AFRL-IF-RS-TR-2005-236
Final Technical Report
June 2005

CASE AND MODEL DRIVEN DYNAMIC TEMPLATE LINKING

BBN Technologies

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. J934

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

STINFO FINAL REPORT

AFRL-IF-RS-TR-2005-236 has been reviewed and is approved for publication

APPROVED: /s/
JOSEPH A. CAROLI
Project Engineer

FOR THE DIRECTOR: /s/
JAMES W. CUSACK, Chief
Information Systems Division
Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	**2. REPORT DATE** June 2005	**3. REPORT TYPE AND DATES COVERED** **Final Mar 00 – Mar 05**

4. TITLE AND SUBTITLE

CASE AND MODEL DRIVEN DYNAMIC TEMPLATE LINKING

5. FUNDING NUMBERS

C - F30602-00-C-0039
PE - 63760E
PR - ATEM
TA - P0
WU - 06

6. AUTHOR(S)

Alice Mulvehill, Michael Callaghan, Mary Kennedy, Rick Altmann, Brian Krisler, Virginia Travers, Ken Anderson

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

BBN Technologies
10 Moulton Road
Cambridge MA 02138

8. PERFORMING ORGANIZATION REPORT NUMBER

N/A

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Defense Advanced Research Projects Agency
3701 North Fairfax Drive
Arlington VA 22203-1714

AFRL/IFSA
525 Brooks Road
Rome NY 13441-4505

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

AFRL-IF-RS-TR-2005-236

11. SUPPLEMENTARY NOTES

AFRL Project Engineer: Joseph A. Caroli/IFSA/(315) 330-4205 Joseph.Caroli@rl.af.mil

12a. DISTRIBUTION / AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 Words)*

This report summarizes a tool created under the DARPA Active Templates (AcT) program that allows users to define templates, actively use (and modify) those templates for problem solving, and supports the reuse of templates based on case-based reasoning. The goal of the AcT program was to develop and apply a spreadsheet-based planning metaphor to improve the Special Operations Forces mission-planning process. The tool, TRACKER – Template Retrieval and Adaptation of Case Knowledge for Evaluation and Reuse, stores and indexes template patterns and employs efficient search and retrieval mechanisms to support the fast retrieval and reuse of templates. TRACKER utilizes techniques from case-based and model-based reasoning to decrease planning time and yield higher quality plans. A tool called the "Browser" was developed to store templates and support the interchange of TRACKER templates with other AcT systems. During the project the TRACKER tool was used to support a number of domains. A version of the tool called Digital Command and Control System (DC2S) was developed to support Special Operations crisis action planning. TRACKER was also modified for application to numerous domains including travel planning, collaborative planning, workflow management, diplomatic clearance requirements for foreign country overflights, and incident response planning after nuclear, biological or chemical attacks.

14. SUBJECT TERMS

Templates, Active Templates, Special Operations Forces, Smart Forms, Mission Planning, Crisis Action Planning, Deliberate Planning, Case-Based Reasoning, Model-Based Reasoning, Collaboration, Work Flow Management

15. NUMBER OF PAGES 60

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Table of Contents

1. Introduction 1
2. Background 1
   - 2.1 Proposed Work 1
   - 2.2 Actual Work 2
     - 2.2.1 The Early Version of the Tool 2
3. What Is a Template? 4
4. What is “Active”? 5
5. Supporting Special Operations Planning 7
   - 5.1 Using Different Views 12
   - 5.2 Building Queries with the Browser 13
   - 5.3 Creating Reports and Customized Views 13
     - 5.3.1 PowerPoint and MS Word Files 13
       - 5.3.1.1 PowerPoint Generation 14
       - 5.3.1.2 Stylized Text File Generation (Warning Order, Fragmentary Order) 14
     - 5.3.2 Using the Form View 14
   - 5.4 Using Email 15
   - 5.5 Connections with Other Systems/Applications 15
   - 5.6 Authoring New Templates 15
   - 5.7 Collaboration 17
     - 5.7.1 Sending to Components 19
   - 5.8 Generating Alerts 22
   - 5.9 Using Scripts and Other Widgets 23
   - 5.10 DC2S Summary 25
6. Other Research Experiments 25
   - 6.1 Workflow Development 25
   - 6.2 Real Time Data Collection and Updates 27
   - 6.3 Experiment with D3i, a Tool Developed by DARPA 28
   - 6.4 CommandLink 29
   - 6.5 DAML 31
   - 6.6 Web Planner 31

6.7 HiCAP 32
6.8 IWARN, An AFRL Incident Reporting Environment 34
6.9 Automated Clearance Tool (ACT) 36
7. General *Tracker* Discussion 36
7.1 Using XML 36
7.1.1 Xpath Work and Scripts 37
7.1.1.1 Slaving Template Fields to Others 37
7.1.1.2 Use of JScheme for Building Scripts 37
7.1.1.3 Use of JScheme to Support Collaboration 38
7.2 Use of Heterogeneous Data Sources 38
7.2.1 Databases 38
7.2.2 Map Files 39
7.2.3 Datafiles 39
7.3 Use of Websites 39
7.4 Use of Images and Graphics 39
7.5 Domains 40
7.6 User Preferences 40
7.7 Built-in Help 40
7.8 Collaboration 41
7.8.1 Email 41
7.8.2 Shared Drive/Server 41
7.8.3 Peer-to-Peer Live Sharing 41
8. The Browser 42
8.1 Indexing Scheme 44
8.2 Fields 44
8.2.1 Node Type 44
8.2.2 Collection 45
8.2.3 Tagname 45
8.2.4 Attributes 45
8.2.5 Content 45
8.3 Optimization 45
8.4 Search Features 46
8.4.1 Find Similar 46
8.4.2 Did You Mean 46

9. Conclusion 47
10. References 48
11. Acknowledgements 49
12. Appendix A 50
    12.1 Example Communication Protocol between *Tracker* and HICAP 50
    12.2 *Tracker*/HICAP Demo 1 51

List of Figures

Figure 1 - Early Travel Program 4

Figure 2 - Early Travel Data Entry Form and Associated Data Templates 6

Figure 3 - Form Generation 7

Figure 4 - Active Template Concept for Operations Order (OpOrd) Development 8

Figure 5 - Communication Protocols Provided for DC2S Application 9

Figure 6 - Crisis Action Planning Checklist 10

Figure 7 - Example Mission Initialization Template 11

Figure 8 - Tasking to Components Template Example 12

Figure 9 - Browser Mission Casebase Example 13

Figure 10 - The Report View 15

Figure 11 - The Modify Properties Menu 17

Figure 12- Collaboration and Updates 18

Figure 13 - Example of How Tasks are Sent to Components 19

Figure 14 - Example Component-Abbreviated-Feasibility-Analysis Template 20

Figure 15 - Example JSOTF-Initial-Mission-Analysis Template 21

Figure 16 - Example Mission Folder Structure; Note Images Folder for Storing Images 22

Figure 17 - Example of the Alert Queue 22

Figure 18 - Basic Tracker Test Template 24

Figure 19 - Basic Tracker Computational Widgets 25

Figure 20 - An Example Workflow with Information Requirements for a Fire Department Plan 27

Figure 21 - The Tracker Version of the Fire Department Workflow 27

Figure 22 - Heracles Weather Data Pulled From a Tracker Template 28

Figure 23 - D3i API 29

Figure 24 - The Tracker/HICAP Concept Overview 33

Figure 25 - Tracker Tax Templates with HICAP suggestions 34

Figure 26 - EAR Templates as Developed in Tracker 35

Figure 27 - Example User Preferences in Tracker 40

Figure 28 - Browser with Query for Ocean against the XML Version of the CIA Fact Book .. 43

List of Tables

Table 1 - DC2S Authoring Widgets 16
Table 2- CommandLink - Tracker Comparison 30

1. Introduction

Tracker is a tool that has been developed by BBN Technologies as part of the DARPA Active Templates program. *Tracker* was created to allow users to define templates, “actively” use (and modify) those templates to support some problem-solving context, and to support the re-use of templates created by *Tracker* or other Active Templates tools. In the course of the project, the tool *Tracker* was used to develop templates that could support a variety of problem areas, including travel planning, tax preparation, Special Operations Forces (SOF) mission planning, and biological-chemical incident data collection. A template “browser” was also developed to support the retrieval and re-use of both templates and the instances that were created in support of a given problem-solving context.

During the course of the program, many experiments were conducted with the tool. Each experiment served to drive extensions and improvements to the *Tracker* tool and the associated browser tool. This paper describes the highlights of several of those experiments. It also highlights the main features of both *Tracker* and the browser.

Our experience to date indicates that *Tracker* can support template development and usage, but it can also be used to define and control workflows, to support the collaborative usage and interchange of templates from a variety of other template generating tools, and to create (author) data entry forms for external applications.

2. Background

2.1 Proposed Work

The goal of the DARPA – AFRL Active Templates (AcT) program was to develop and apply a spreadsheet-based planning metaphor to improve the Special Operations Forces (SOF) mission-planning process. BBN proposed the development of a tool that would leverage features of both case-based reasoning (CBR) and model-based reasoning. The tool, named *TRACKER* – Template Retrieval and Adaptation of Case Knowledge for Evaluation and Reuse (henceforth, *Tracker*), was intended to store and index template patterns and to employ efficient search and retrieval mechanisms to support the fast retrieval and reuse of templates. The tool was intended to include methods that would support the automatic linkage of templates defined within *Tracker* to other templates and/or external processes. The tool would support two types of users: administrative users in the development of templates that would serve as master templates, and end users who could use the master templates to solve some problem. The tool would also enable the users (both administrative and end users) to modify templates and to attach new remote processes to them as needed.

All of the tools developed in the AcT program were intended to support SOF mission planning. SOF commanders and their staff use manual means to implement complicated deployment and employment decision processes, to force coordination, and to make plan revisions. Time is consumed searching for, manipulating, and sharing information, thus reducing the allotted time for making difficult decisions. Using techniques from case-based and model-based reasoning, active-template tools were intended to decrease planning time and yield higher-quality plans. The vision was for SOF users to modify pre-defined model templates to rapidly assemble mission plans that are based on operational orders, while accommodating continuously changing requirements and unforeseen constraints.

Tracker was intended to support the development, indexing, storage, retrieval, reuse, modification, and merging of templates. Subject matter experts (SMEs) from the SOF community would be consulted to help define a case repository (casebase) of SOF missions. The *Tracker* tool would be used to support the user during plan development. The vision was that templates associated with SOF missions would be linked to other kinds of software systems, including information systems and databases, rule and constraint-based systems, domain models, constraint satisfaction and problem-solving systems, software agents, and other special-purpose reasoning tools.

2.2 Actual Work

During the project we were able to achieve many of our research objectives. The *Tracker* tool was used to support the SOF community through the specialized version of *Tracker* called DC2S (Digital Command and Control System). A tool called the “Browser” was developed to store templates and mission instances (missions developed with the use of the templates). The DC2S system was able to access other software systems and databases. Additionally, the DC2S system was demonstrated during the Millennium Challenge 2002 exercises. Section 5 of this paper describes many of the features of this application.

Although the SOF community did not see a critical need for the casebase (for reuse of past plans), as part of this project, we did develop the browser tool to support case-based reasoning for a variety of other problems. For example, the initial version of the browser was developed to support travel planning. This simple browser is now part of *Tracker* and any derivative *Tracker* application. Later a more powerful browser was developed to support the interchange of templates with other Active Templates systems such as D3i. This later browser (described in Section 8) was transferred to another DARPA program called JAGUAR, where it is used today to store executed plans and models as casebase repositories.

2.2.1 The Early Version of the Tool

One of the primary goals of the Active Templates (AcT) program was to develop tools that were essentially free-form, but not unstructured; that could be used by the end user as a product and could also be used by a software designer to author forms/templates. Tools developed by the program were intended to provide an expressive structural descriptive power that could be changed at any time; that could support the re-use of structural descriptions and past values, and that were visually attractive and easy to work with.

The tools developed during the program were intended to manipulate and share structures called “templates”. The templates were intended to provide more than passive interactions between a user and the software; they would be “Active” (hence Active Templates).

Early in the AcT program each research team was directed by the program manager to build an active templates tool prototype within 100 days. For BBN, our 100-day prototype development goals were:

- Build a set of templates that *support hierarchical planning*. Focus on at least two sub-templates, e.g., synchronization of resources and asset management.

- Build a tool that is capable of using *heterogeneous data* sources supporting different use cases, e.g.,

    1. Information pulled from open-source Internet.

    2. Information pulled from local data sources.

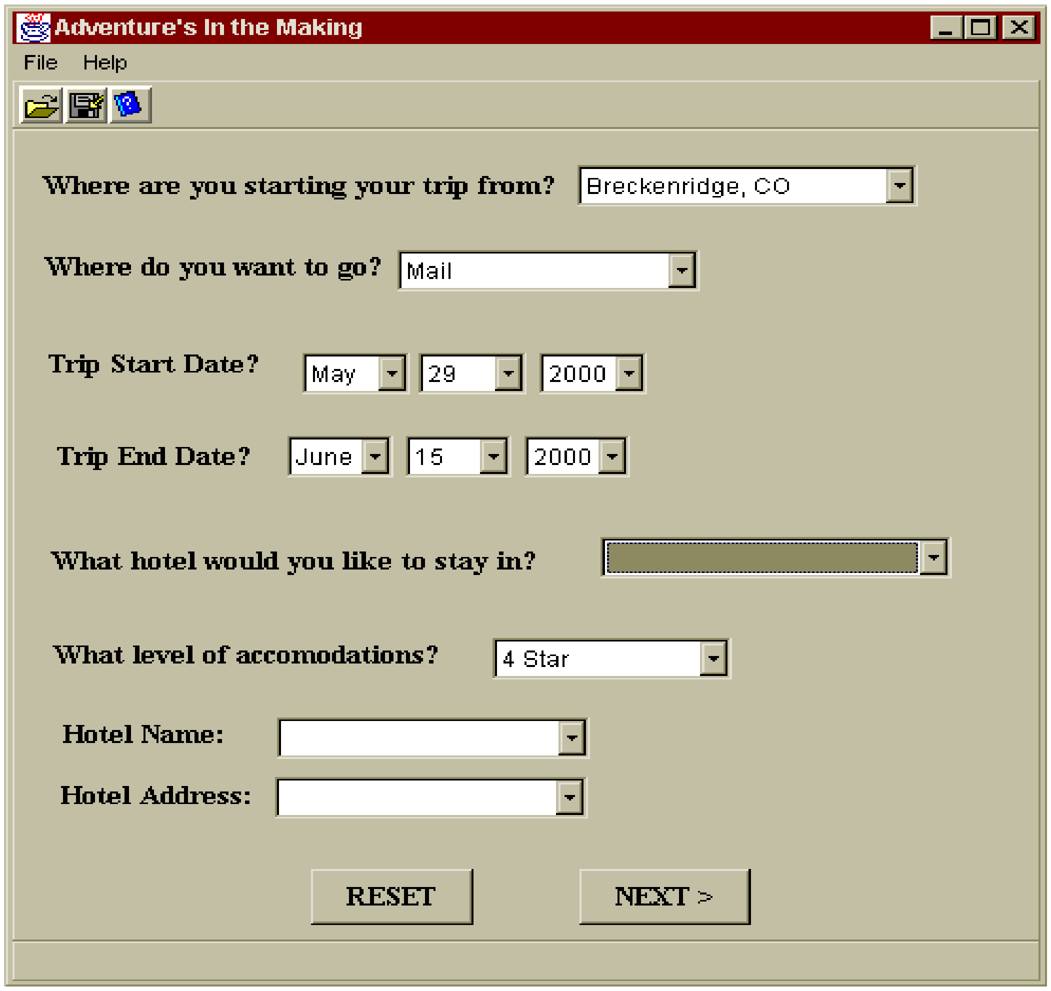
    3. Information pulled from local LAN data sources.

    4. Information pulled from other software sources.

The first BBN effort focused on the development of a template-based tool to support travel planning. We chose this domain because it presented a myriad of challenges similar to what military troops face when they deploy to a remote location, e.g., setting up departure/arrival locations and times. In addition, the travel domain offered a planning problem that was unclassified and that required little acquisition of domain data. The travel domain also offered us the opportunities to address the goals we set for the 100-day effort. Figure 1 shows the early user interface of the tool that was produced to satisfy the 100-day objectives. The tool, called *Tracker,* provided the user with a set of templates that were designed in Java [1] by a Java programmer. A user could then use these templates to enter data about a trip. The tool would store the trips in a PostgreSQL database (www.postgresql.org) and the values stored in this database could be re-used to provide values for similar trips. For example, in Figure 1, a trip is being planned to go to a place called "Mail" from Breckenridge, Colorado. Once this trip is created, it is stored in the database. If the traveler needs to take another trip from Breckenridge, Colorado to this destination, the relevant travel information from the stored trip would be retrieved from the database, thereby allowing the user to re-use the information from the previous trip. For example, the tool could help the user make a hotel reservation by retrieving data about the hotel that was used on previous trips and using the retrieved data about that hotel for the new trip.

In order to specify data for an entire trip, the user of this first interface had to rely on the "next" button (as displayed in Figure 1). This button would step the user through an ordered set of pre-defined templates that modeled the trip planning process. Each template presented questions to the user about their need for hotels, car rentals, airplane reservations, and other data of relevance to traveling. This early version of *Tracker* provided links to external data sources, like the Internet, in order to avail the user of related travel data such as weather forecasts and maps. The tool also stored user preferences that the travel templates could use for default preferences, such as requests for a non-smoking hotel room.

Although we were able to support travel planning with this early *Tracker* application and to achieve many of our 100-day objectives, our work unveiled a number of research issues that would fuel our subsequent tool development and research efforts. These research issues included problems associated with: partial planning, the merging of plans, the validation of the data sources that were used to populate values in instantiated templates, methods for determining the confidence of the user in the data sources, indexing methods that could facilitate the retrieval of data for re-use in new plan or template construction, and a need to enable the user to develop and extend templates directly.



Where are you starting your trip from? Breckenridge, CO

Where do you want to go? Mail

Trip Start Date? May 29 2000

Trip End Date? June 15 2000

What hotel would you like to stay in?

What level of accomodations? 4 Star

Hotel Name:

Hotel Address:

Figure 1 - Early Travel Program

3. What Is a Template?

From a data-entry perspective, a template is a form (frame) with slots that can be filled in to describe or model some entity (process or object). The slots of the template are composed of attribute-value pairs that describe some object or some set of actions associated with a task or process. For example, a template can be used to describe a person, with each attribute specifying data such as: first name, last name, age, date of birth, etc. When the values of the attributes are filled in to describe some particular person (personX) we refer to that set of data values as an instance.

A template can be comprised of other templates. For example, a travel template can be comprised of a person template that describes the person who is traveling, a hotel template that describes a hotel, a hotel reservation template that describes a transaction between a person and a hotel to allow the person to stay at a room in the hotel for some time period, and a variety of other templates that describe other aspects of travel, e.g., a car-reservation template.

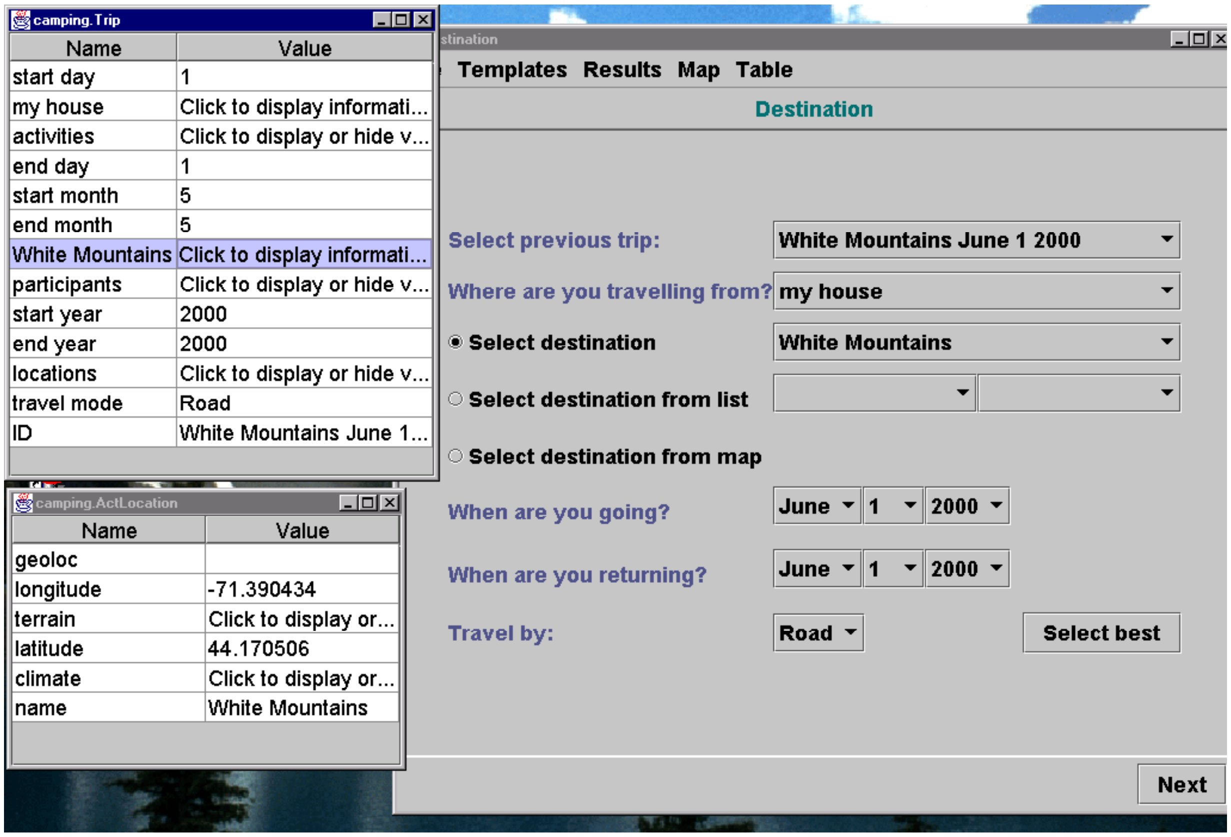
From a more abstract perspective, a template can also be viewed as a pattern that is associated with a class of problems. Each template has features that make it a member of a class of templates; that distinguish it from other patterns and features; and that allow it to link (via it's slots) to other patterns. Templates, when designed within a problem-solving context as a reference, can simplify problem solving in that context by (a) providing flexible ways to make and record decisions, (b) reminding the user to perform certain tasks, (c) encapsulating experience, and (d) constraining task specification and language.

Templates are commonly used to support data entry that is associated with some process. Data-entry templates are often referred to as "forms". For our work, we use "form" and "template" as interchangeable terms. The United States tax forms are a set of well-known forms that have been created to help people compute their taxes. The forms or detailed templates that make up the IRS tax forms describe tasks or decision points and their alternatives. There is the main 1040 form, and then many schedules (detail templates) that are filled in as needed. The tax templates that make up the IRS tax forms are effective because each tax form is designed to collect data relative to some aspect of taxation as defined in a model that embodies the laws governing tax collection. Programs like TurboTax [2] use the tax model, the data elements of the templates, and information about the person who is being taxed to determine what forms need to be filed. Depending on the financial situation of a person, one form (the EZ form) or many forms are required in order for that person to file their taxes. Programs such as TurboTax enhance the paper process by supporting the propagation of data values across fields as appropriate. For example, in TurboTax, the name of the person filing the tax and their social security number, once entered, is automatically propagated to all similar name and social security data entry fields. In addition, programs such as TurboTax also provide some computation services, e.g., adding up the values of certain fields to generate the value for another field.

While a program like TurboTax supports the tax-problem-solving domain with a set of templates that users can use to compute and file their taxes, the *Tracker* tool, developed for the Active Templates program, does three other tasks: it uses the notion of a "domain" to support usage across multiple problem-solving contexts; it supports the authoring of the templates; and it supports template evolution – allowing the usage of templates (the template instances) to influence the update of the master template. For example, if an end user could add another line item to the tax form, *Tracker* can use this new field to update the master tax template. Although that might not be desirable in a problem-solving context like taxation, it is a very valuable feature for other data-entry problems, like those associated with the recording of biological-chemical incidents [3].

4. What is "Active"?

One of the overarching goals of the Active Templates program was to develop templates that went beyond static forms. The goal was to develop the templates so that they were "active". The "active" trait of the templates could take a variety of forms. For example, a slot could update itself when the value of a related slot changed, e.g., as when computing a value that is based on other input values. Another example is the auto-population of a slot or of an entire template based on the entry of a certain value. For example, in the travel templates, when PersonX is traveling to a specific location, other data services, such as a geographic-location database can be accessed to provide the latitude and longitude of the destination. In our early work, a user could specify where they wanted to go through the interface, and the templates associated with that interface would be updated. For example, Figure 2 shows the interface along with two of the data-entry templates as they are being specified via the interface and via calls to other services. This figure also displays a button called “select best”. If the user pressed this button, data from the tool was passed to another tool called XpertRule [4] where it was used to compute the best mode of transportation.



camping.Trip

Name	Value
start day	1
my house	Click to display informati...
activities	Click to display or hide v...
end day	1
start month	5
end month	5
White Mountains	Click to display informati...
participants	Click to display or hide v...
start year	2000
end year	2000
locations	Click to display or hide v...
travel mode	Road
ID	White Mountains June 1...

camping.ActLocation

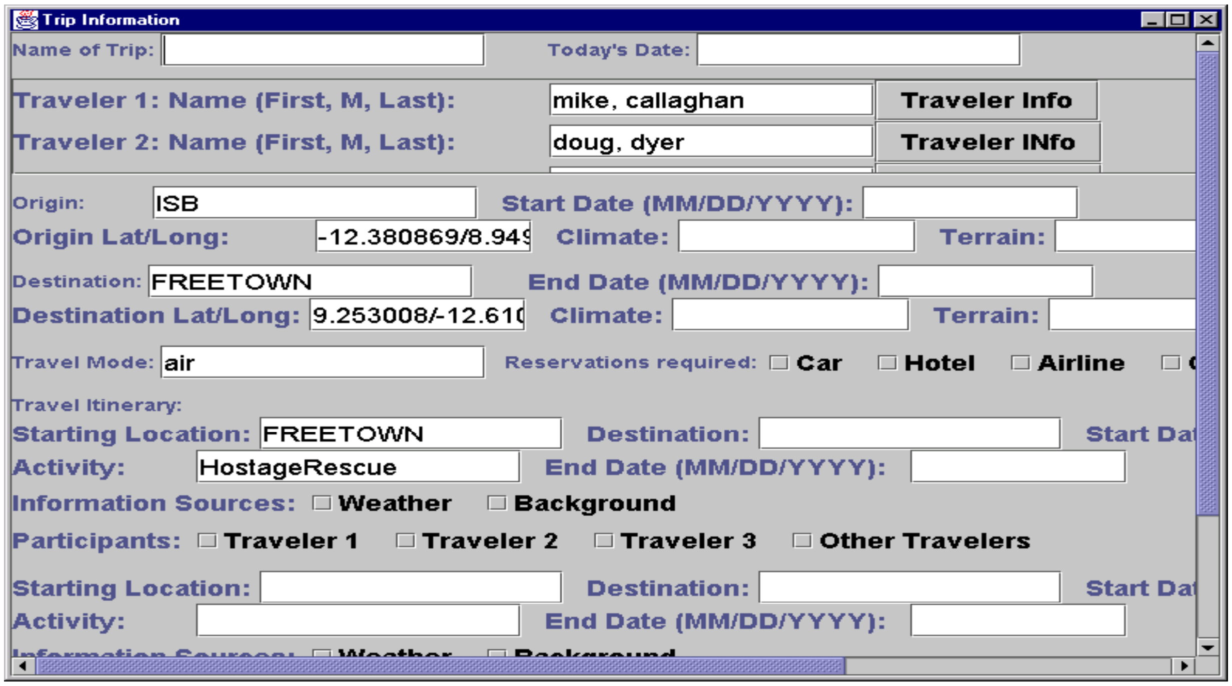
Name	Value
geoloc	
longitude	-71.390434
terrain	Click to display or...
latitude	44.170506
climate	Click to display or...
name	White Mountains

Select previous trip: White Mountains June 1 2000
Where are you travelling from? my house
● Select destination: White Mountains
○ Select destination from list
○ Select destination from map
When are you going? June 1 2000
When are you returning? June 1 2000
Travel by: Road

Figure 2 - Early Travel Data Entry Form and Associated Data Templates

XpertRule is an inductive inference engine with an Application-Program Interface (API) that allows it to communicate with external programs. We used XpertRule to determine which mode of travel a person should take to get from one location to another. The idea was that we would use the templates to collect the data, and then pass relevant data to another program that would make the computation. This would be analogous to the tax computation programs that are accessed by tax-preparation programs. While the idea was good, the API between *Tracker* and the XpertRule program proved to be difficult to use. But the ultimate determination to suspend usage of XpertRule was a decision by the DARPA program manager to not use third party software that required a license. Hence we stopped our usage of XpertRule.

By the end of the first year, we began to concentrate less on travel planning and more on SOF mission planning. We began by generating forms that could display the data collected through the interface and stored in the templates. Figure 3 is an example of this form. It shows values that have been collected through data entry, values derived through calls to database sources such as the GEOLOC (geographic-location) database (for the values of Latitude and Longitude), values set by reference to user preferences (airline, hotel), and values derived through the Internet, e.g., weather. In order to more automatically provide data to the forms, we developed Java code to utilize an Xpath [5] specification as a data-access interface to the XML structure underlying our templates. This approach proved useful, so we started to rely less on databases for storage and instead to make almost exclusive use of XML file storage. Work with Xpath also resulted in the birth of the “browser”.


Trip Information

Name of Trip: Today's Date:

Traveler 1: Name (First, M, Last): mike, callaghan Traveler Info
Traveler 2: Name (First, M, Last): doug, dyer Traveler INfo

Origin: ISB Start Date (MM/DD/YYYY):
Origin Lat/Long: -12.380869/8.94 Climate: Terrain:

Destination: FREETOWN End Date (MM/DD/YYYY):
Destination Lat/Long: 9.253008/-12.61 Climate: Terrain:

Travel Mode: air Reservations required: ☐ Car ☐ Hotel ☐ Airline ☐

Travel Itinerary:
Starting Location: FREETOWN Destination: Start Da
Activity: HostageRescue End Date (MM/DD/YYYY):
Information Sources: ☐ Weather ☐ Background
Participants: ☐ Traveler 1 ☐ Traveler 2 ☐ Traveler 3 ☐ Other Travelers

Starting Location: Destination: Start Da
Activity: End Date (MM/DD/YYYY):

Figure 3 - Form Generation

The early browser was intended to help a user to find templates that described characteristics about a certain entity, e.g., the skills of a particular traveler. This early browser used an Xpath specification to allow the user to make queries against stored trips (stored as XML files) based on attributes such as destinations, dates, activities, participants’ names, skills, etc.

By the end of the first year of the project we were able to support planning with a set of templates and data reports. Although we were able to propagate values from the templates to the reports, we still relied on the reports to be externally created. The next version of *Tracker* added the ability of the user to author the reports (as reports are just another type of template) from scratch; import templates from files, URLs, and database records; link external files, URLs, and database records; and interchange templates with other developers as email attachments. In the new version of *Tracker*, all non-database data is stored as XML. Because this new version was intended to replace the application built to date, the new application also supported a capability to read and re-format the XML data that had been generated in the previous version.

5. Supporting Special Operations Planning

With the new version of *Tracker*, we were positioned for a domain application that would test and force the tool to evolve. In the 2nd year of the program we were introduced to a group of military planners from the Army Special Operations Battle Laboratory (ARSOBL).

Interactions with the users and subject matter experts (SMEs) resulted in the generation of a specialized version of *Tracker* that is called Digital Command and Control System (DC2S).

In our research we assume that templates, when filled in and linked, can represent entire plans. When the assumption is extended to the Special Operations Forces (SOF), we would expect that these users will start with a general *model template* for the kind of operation envisioned, and then selectively link in other templates that serve not only as reminders of what is required for a particular type of plan, but as indicators of data-collection requirements and problems or issues to be addressed. At the start of our work with ARSOBL, we envisioned the SOF users modifying pre-defined model templates to rapidly assemble detail templates, and in the process receiving warnings of changing requirements and circumstances (See Figure 4). We assumed that the SOF users might want to reuse template information from previous plans, shown in Figure 4 as the Case-Based Reasoning (CBR) element. We also assumed that they might want to relax the constraints associated with templates to handle new exceptions and to rapidly respond to changes in the environment and/or to maintain an element of surprise while not losing sight of plan dependencies.

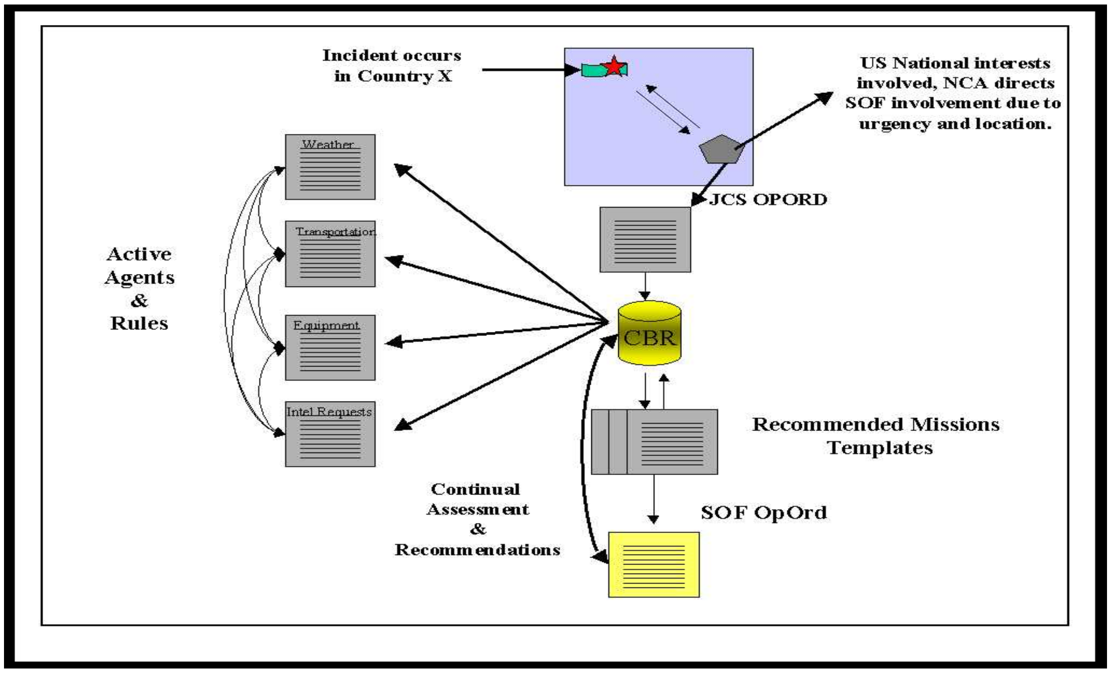
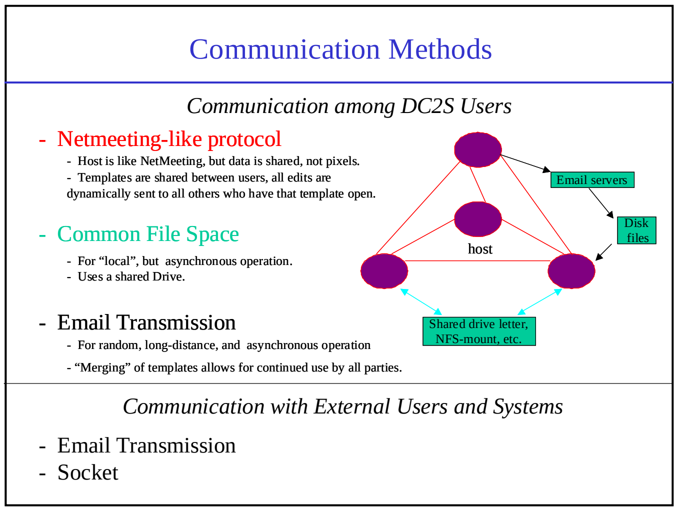

Incident occurs in Country X
US National interests involved, NCA directs SOF involvement due to urgency and location.
JCS OPORD
Weather
Transportation
Equipment
Intel Requests
Active Agents & Rules
CBR
Recommended Missions Templates
SOF OpOrd
Continual Assessment & Recommendations

Figure 4 - Active Template Concept for Operations Order (OpOrd) Development

We spent over one year working with SMEs from ARSOBL to develop DC2S. Through several intense knowledge-acquisition sessions we were introduced to the deliberative planning methods employed by the Special Operations Forces. A document (an Excel spreadsheet) provided by ARSOBL defined the overall model and workflow associated with their deliberate planning process. This document served as the basis for the development of a variety of templates specially crafted to support the many special operations missions. Using this document, and with the help of the ARSOBL staff for interpretation, we were able to develop a set of templates to support the planning required.

A number of special features were developed to support DC2S. Although we had developed special features to support problem-solving domains we had encountered earlier, it was not until the experience with the DC2S application that we were faced with the need to develop an explicit “domain” concept that would support the loading of problem-specific features of relevance into *Tracker*. For DC2S, this included a set of tools that allowed us to compute the completeness of a set of forms, to generate maps in the map tool C2PC [6], to develop specialized collaboration tools, to load in and manage a checklist, and to generate a set of PowerPoint slides that incorporated the data as collected in the various mission templates.

During the work on the DC2S application a large number of new features were added to *Tracker*. Many of these features are described in Section 7 of this paper. Although we developed templates to support deliberate planning, we also built templates to support Crisis Action Planning (CAP). Because of the near-real-time aspect of crisis action planning, we introduced a peer-to-peer collaboration capability in DC2S. Figure 5 describes the communication schemas available in *Tracker*/DC2S to support collaborative template authoring or usage.


Email servers
Disk files
host
Shared drive letter, NFS-mount, etc.

Communication Methods

Communication among DC2S Users

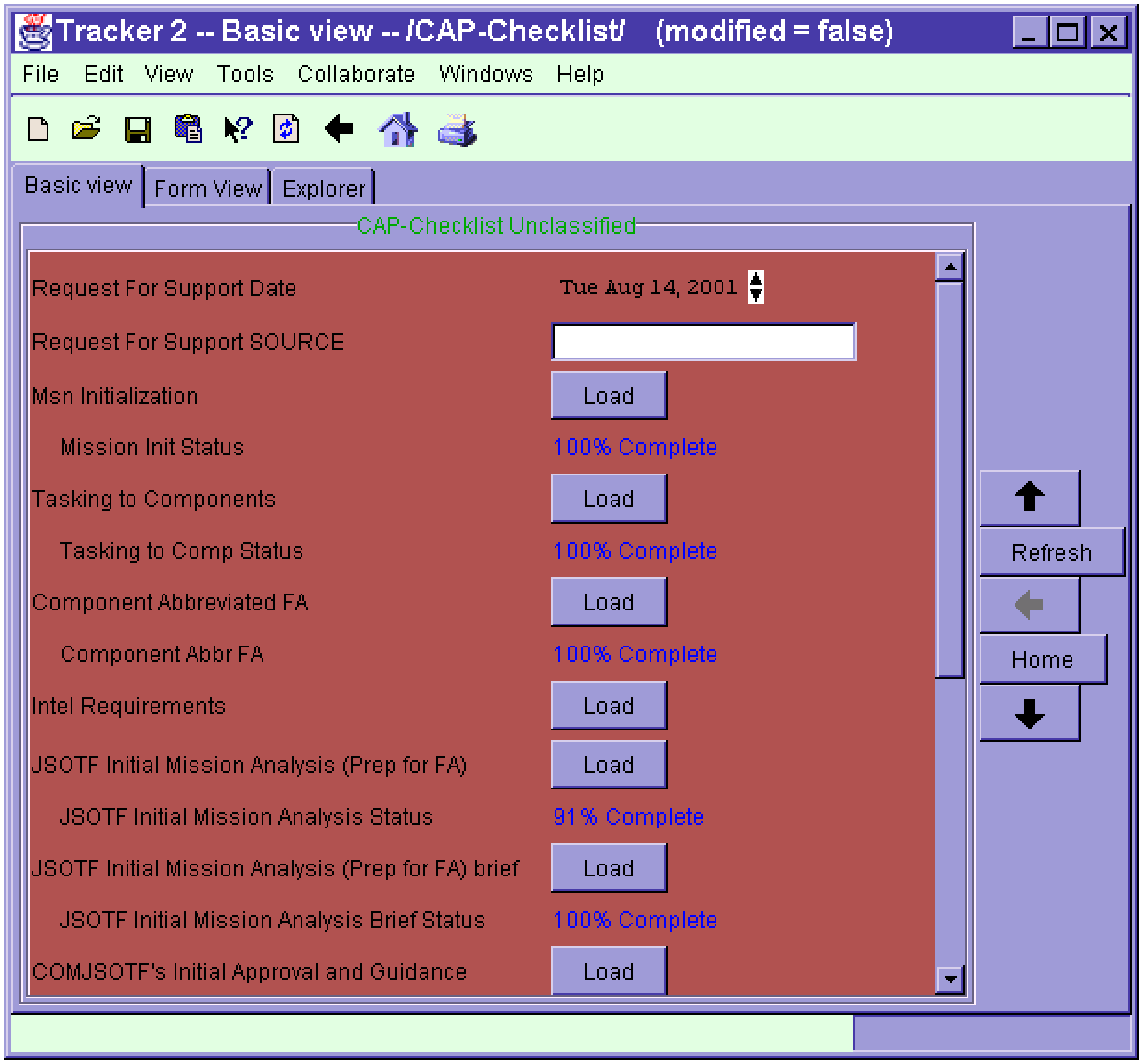
- Netmeeting-like protocol
  - Host is like NetMeeting, but data is shared, not pixels.
  - Templates are shared between users, all edits are dynamically sent to all others who have that template open.
- Common File Space
  - For “local”, but asynchronous operation.
  - Uses a shared Drive.
- Email Transmission
  - For random, long-distance, and asynchronous operation
  - “Merging” of templates allows for continued use by all parties.

Communication with External Users and Systems

- Email Transmission
- Socket

Figure 5 - Communication Protocols Provided for DC2S Application

In order to better manage the crisis action planning process we developed a checklist. The checklist serves as a type of workflow, allowing the user(s) to selectively load templates and track their completion. An example is provided in Figure 6. When all of the “required” slots of the template are specified and saved, the percent-complete indication on this checklist will change to display 100%.

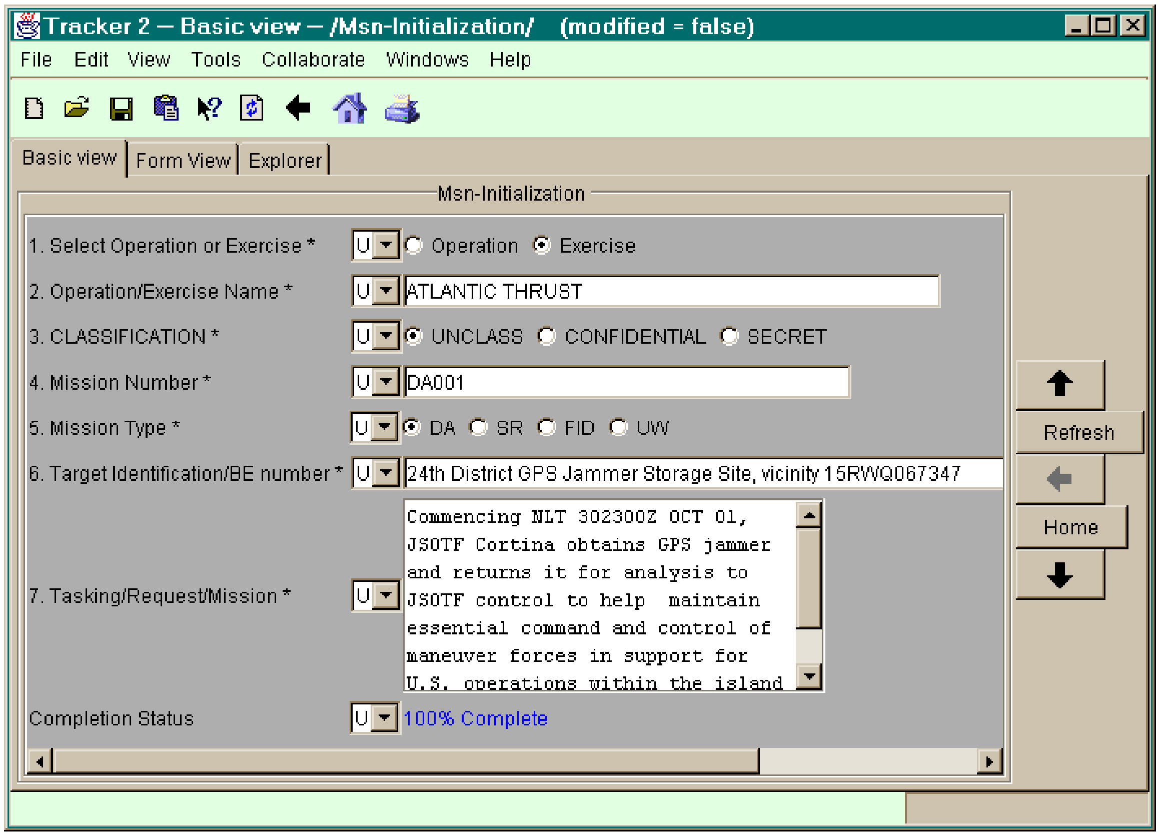

Tracker 2 -- Basic view -- /CAP-Checklist/ (modified = false)

CAP-Checklist Unclassified

Request For Support Date	Tue Aug 14, 2001
Request For Support SOURCE	
Msn Initialization	Load
Mission Init Status	100% Complete
Tasking to Components	Load
Tasking to Comp Status	100% Complete
Component Abbreviated FA	Load
Component Abbr FA	100% Complete
Intel Requirements	Load
JSOTF Initial Mission Analysis (Prep for FA)	Load
JSOTF Initial Mission Analysis Status	91% Complete
JSOTF Initial Mission Analysis (Prep for FA) brief	Load
JSOTF Initial Mission Analysis Brief Status	100% Complete
COMJSOTF's Initial Approval and Guidance	Load

Figure 6 - Crisis Action Planning Checklist

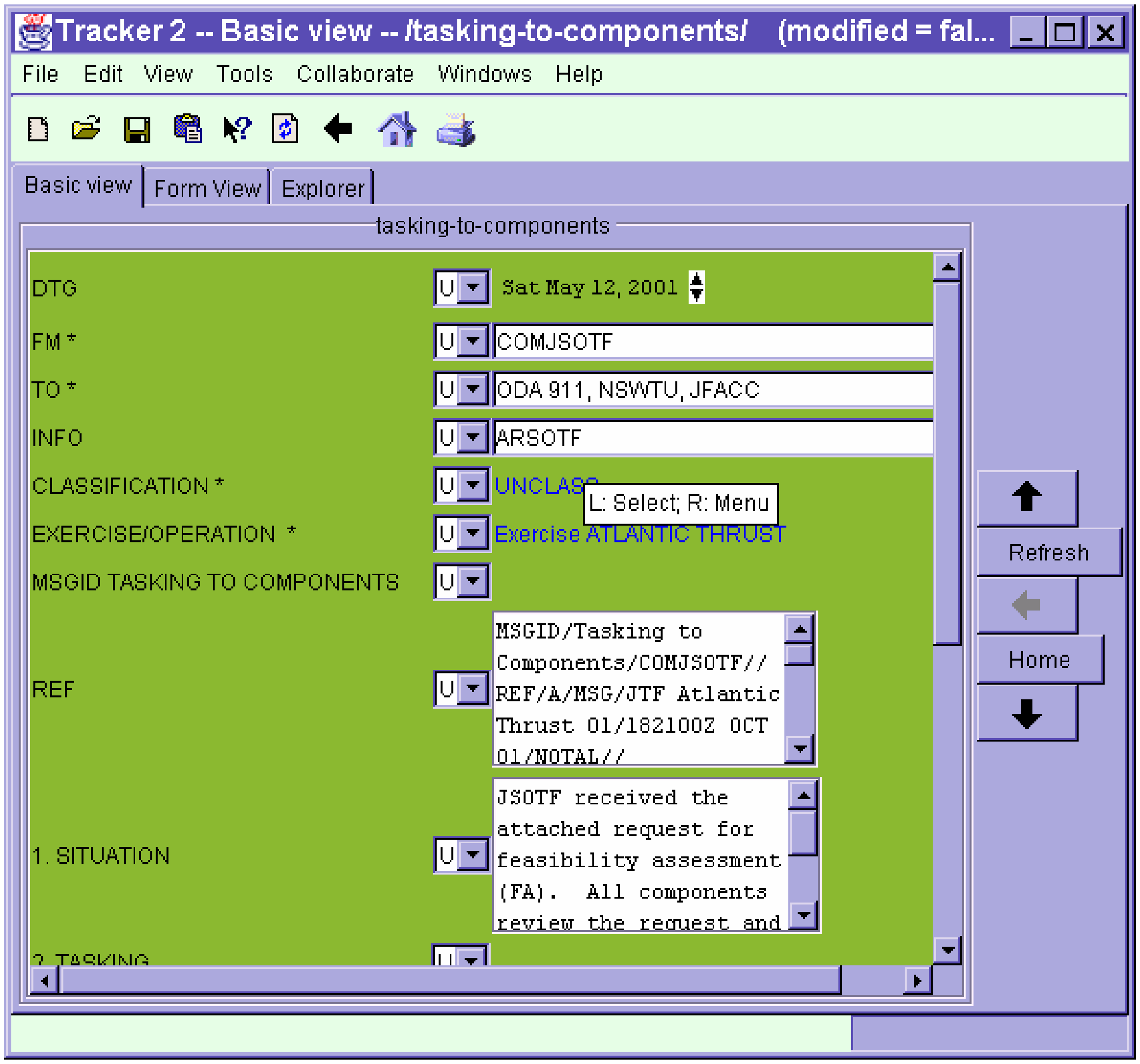
For crisis action mission planning, the first template that needs to be loaded is the template that defines the purpose of the mission or basically initiates a mission. Figure 7 shows an instantiated Mission Initialization template. Note that most of the slots/fields in this template are annotated with an asterisk (*). In DC2S, this asterisk is used to indicate that the user must provide data for that slot/field. The template displayed in Figure 7 has been completely specified, as indicated by the "completion status" of 100%. Note that the classification of each of the fields can also be specified. Note that this template contains a variety of field types, to include text line, text field, radio button, and computation fields.


Tracker 2 — Basic view — /Msn-Initialization/ (modified = false)
File Edit View Tools Collaborate Windows Help
Basic view
Form View
Explorer
Msn-Initialization
Refresh
Home

1. Select Operation or Exercise *	U	Operation [✓ Exercise]
2. Operation/Exercise Name *	U	ATLANTIC THRUST
3. CLASSIFICATION *	U	[✓ UNCLASS] CONFIDENTIAL SECRET
4. Mission Number *	U	DA001
5. Mission Type *	U	[✓ DA] SR FID UW
6. Target Identification/BE number *	U	24th District GPS Jammer Storage Site, vicinity 15RWQ067347
7. Tasking/Request/Mission *	U	Commencing NLT 302300Z OCT 01, JSOTF Cortina obtains GPS jammer and returns it for analysis to JSOTF control to help maintain essential command and control of maneuver forces in support for U.S. operations within the island
Completion Status	U	100% Complete

Figure 7 - Example Mission Initialization Template

If we look at another of the templates available through the CAP checklist (as displayed in Figure 8), we will see an example of how a value that was specified in the Mission Initialization template is propagated to another related form. The values filled in and colored BLUE (CLASS and EX/OP) were inherited from the Msn-Initialization template. The propagation of values is achieved through a technique that we call *slaving* (see section 7.1.1.1 for more details.



DTG U Sat May 12, 2001
FM * U COMJSOTF
TO * U ODA 911, NSWTU, JFACC
INFO U ARSOTF
CLASSIFICATION * U UNCLASS
EXERCISE/OPERATION * U Exercise ATLANTIC THRUST
MSGID TASKING TO COMPONENTS U
REF U MSGID/Tasking to Components/COMJSOTF// REF/A/MSG/JTF Atlantic Thrust 01/182100Z OCT 01/NOTAL//
1. SITUATION U JSOTF received the attached request for feasibility assessment (FA). All components review the request and
2. TASKING U

Figure 8 - Tasking to Components Template Example

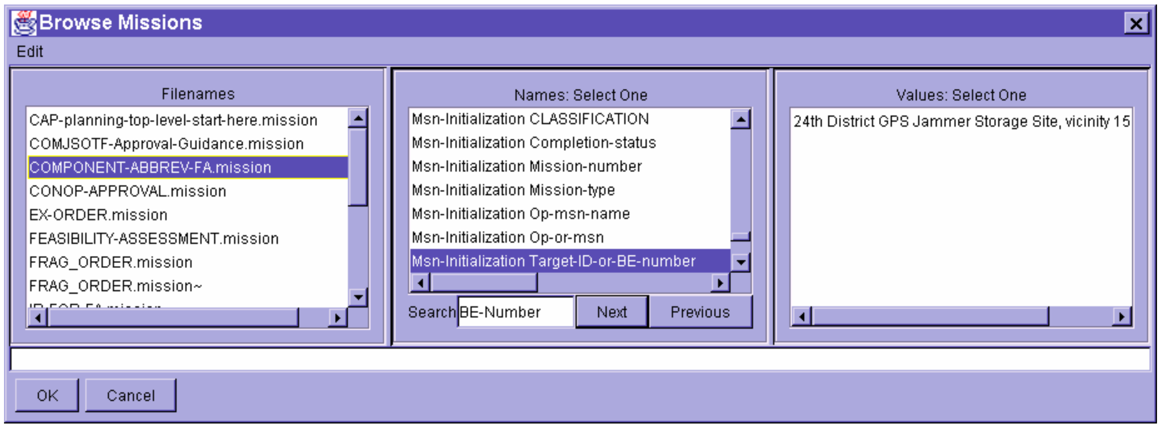
5.1 Using Different Views

To support DC2S as well as some of our concurrent research interests, we enhanced *Tracker* to support three different template *views*. Figures 8 show the template in the "Basic View". The Basic View simply displays the attribute-value pairs of the template as fields with values. The user is expected to use the arrows or buttons to navigate across templates. Another view is the "Explorer View" (See Figure 19 for an example). While similar, it provides a tree view (like the nested folders and documents you find in "Windows Explorer"). The user can navigate by clicking on the folders in the tree. The third view is the "Form View". This view is intended to display fields in some particular spatial configuration. See Figure 10 for an example.

An additional feature of the Explorer View is the search field/controls above the folder-tree. The content of the typed-in field are used to search the content of the loaded XML template or instance.

5.2 Building Queries with the Browser

The browser capability was enhanced to support the DC2S application. While some searching for a field or value is available in the Explorer View, the user can search for a field or value that could be stored in any template (or for DC2S, an entire mission folder or a folder of mission folders). In Figure 9, the user has entered a search request, (BE-Number), and the browser returns the template that supports the specification of a BE-Number.



CAP-planning-top-level-start-here.mission
COMJSOTF-Approval-Guidance.mission
COMPONENT-ABBREV-FA.mission
CONOP-APPROVAL.mission
EX-ORDER.mission
FEASIBILITY-ASSESSMENT.mission
FRAG_ORDER.mission
FRAG_ORDER.mission~

Msn-Initialization CLASSIFICATION
Msn-Initialization Completion-status
Msn-Initialization Mission-number
Msn-Initialization Mission-type
Msn-Initialization Op-msn-name
Msn-Initialization Op-or-msn
Msn-Initialization Target-ID-or-BE-number

Search BE-Number

24th District GPS Jammer Storage Site, vicinity 15

Figure 9 - Browser Mission Casebase Example

The user can also press the button called *Next* to continue searching. Each time that a search succeeds, the value of the search will be presented in the window to the right. For example, in Figure 9 the highlighted value in the center window corresponds to the field that contains the value *BE-Number*. The value in the right window is the complete value of that field, and the highlighted value in the left window is the template in which the field/value exists. The user can click on the template name in the left window to open the template directly from the browser window. With this capability, the user has the ability to use, revise, or merge the imported template or instance

5.3 Creating Reports and Customized Views

In earlier versions of *Tracker*, a "report" was a customized form that was developed by a Java programmer that could be populated with information from the *Tracker* templates. In DC2S, a report can be formed from any template by changing its view to be a "Form View" or through three types of custom reports: a PowerPoint brief, an ASCII text file, or a MS-Word document. In DC2S, a PowerPoint brief could be generated from the template content to brief the mission situation, and a Warning Order could be generated from the template content in the required military format to specify mission requirements to other military components.

5.3.1 PowerPoint and MS Word Files

For many planners, the PowerPoint brief is a common way to communicate plan status. By the end of our DC2S development period, we were able to generate a PowerPoint brief in a format requested by ARSOBL. The PowerPoint brief could be further edited to display specific logos or color/layouts. The user could generate a Warning Order (in its preferred format) by simply pressing the appropriate Load button off the CAP-Checklist. When this button is pushed, a template containing the data for the Warning Order appears. The user can then save the data as

ASCII output or specify a file name for the MS format. The Warning Order could then be emailed to any subordinate commands.

5.3.1.1 PowerPoint Generation

For any template, it is possible to generate a PowerPoint file, where each slide contains one data field (except when that field is a table, and then because PowerPoint can create a table, that slide will have a table). An image field will have the image inserted on the slide. All other field types appear in the slide as text.

PPT slides were generated to support DC2S with a separate program that is written in Visual Basic (VB). This program has clean and simple access to PowerPoint's DLL libraries. *Tracker* can call this VB program, which does most of the work to convert the content from a *Tracker* template into one or more PPT slides.

There are four special PowerPoint-template (.pot) files; one is the title slide, one handles text fields, one handles image fields, and one handles table fields. These PPT template files are fragile due to an internal field-object-identification mechanism that the VB program and DLLs must use. PowerPoint itself must be installed on the machine where it will be used.

A resulting PowerPoint presentation file can be viewed and modified to satisfy the desired presentation style, e.g., logo and layout of the user. In addition, some amount of tune-up is often required, especially for table data, because large tables often overflow the slide borders. Note: The PowerPoint-generation feature was specifically developed to support the DC2S effort. Many versions of *Tracker* do not have this feature as a standard capability.

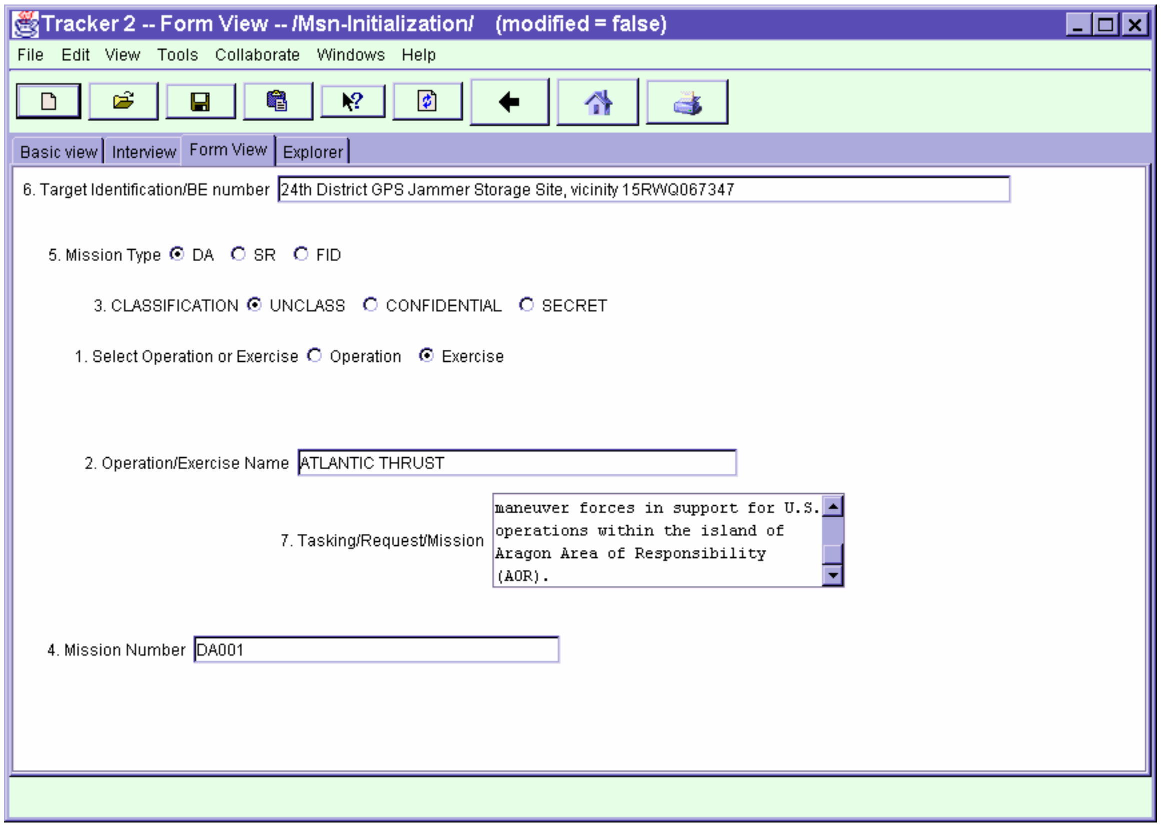
5.3.1.2 Stylized Text File Generation (Warning Order, Fragmentary Order)

Because a template field can be a custom Java widget, that widget can do anything Java can do. This can include generating a special output file based on the field values of the template.

For DC2S, this was done for two templates, the Warning Order and the Fragmentary (Frag) Order. Note: This feature was specifically developed to support the DC2S effort. Many versions of *Tracker* do not have this feature as a standard capability.

5.3.2 Using the Form View

In *Tracker*/DC2S, any template can be changed to a form by simply clicking the Form View tab and then clicking and dragging a field/value to the location of choice. Figure 10 shows an example. While this feature was available in DC2S, the users of DC2S tended to use the Basic View.


Tracker 2 -- Form View -- /Msn-Initialization/ (modified = false)
File Edit View Tools Collaborate Windows Help
Basic view Interview Form View Explorer

6. Target Identification/BE number: 24th District GPS Jammer Storage Site, vicinity 15RWQ067347

5. Mission Type [✓ DA] SR FID

3. CLASSIFICATION [✓ UNCLASS] CONFIDENTIAL SECRET

1. Select Operation or Exercise: Operation [✓ Exercise]

2. Operation/Exercise Name: ATLANTIC THRUST

7. Tasking/Request/Mission: maneuver forces in support for U.S. operations within the island of Aragon Area of Responsibility (AOR).

4. Mission Number: DA001

Figure 10 - The Form View

5.4 Using Email

Tracker/DC2S allows the user to email a template. Email was the method of choice in DC2S for communicating Warning Orders or other order documents to subordinate commands.

5.5 Connections with Other Systems/Applications

DC2S has been configured to run with other applications and/or systems, however, DC2S did not directly make use of this capability. Instead, collaboration methods as discussed later in this paper were preferred.

5.6 Authoring New Templates

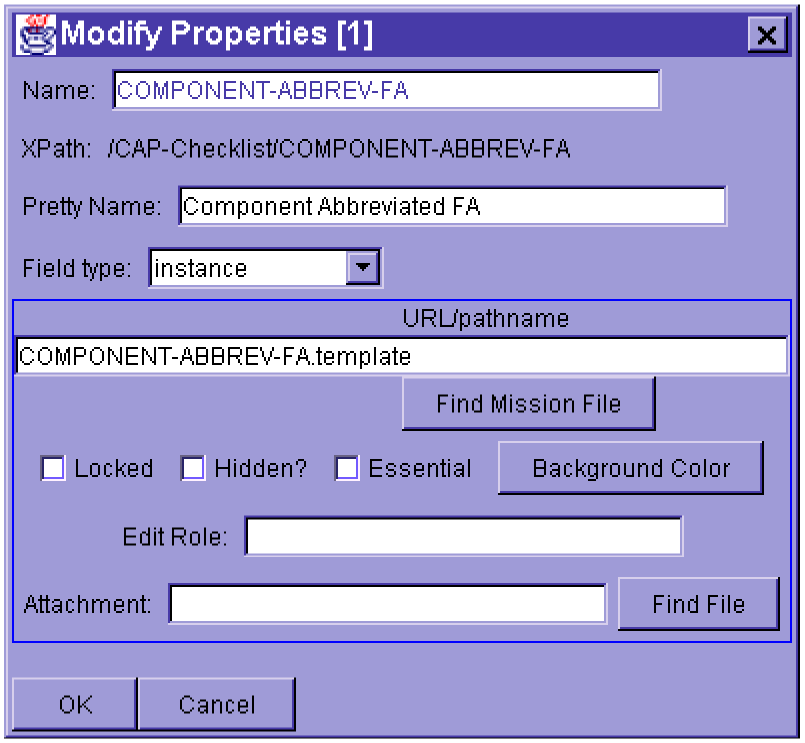
Our goal in DC2S was to support template authoring directly, removing the need for any form/template to be developed in Java by a programmer. The development of the DC2S templates prompted us to define a set of authoring features or widgets. The following table shows a list of the features that were available in DC2S. The final version of *Tracker* contains additional tools.

While these edit tools are available via pull down menus, the system can be set in the “author mode,” where buttons that correspond to the short names in the following list are provided.

Table 1 - DC2S Authoring Widgets

Button Label	***Explanation***
TF:	Text Field. Contains one line of text.
TA:	Text Area. Contains multiple lines of text.
Lbl:	Label. Contains one line of text that is not changeable by a user.
Sldr:	Slider. Can be used to set a numeric value over a hi/lo range.
CB:	Checkbox. Allows the user to specify true/false value.
DB:	A database field. Imports a database record as though it were a sub-template.
RB:	Radio buttons. Allows for the choice one of several pre-defined values.
Date:	Date, displayed attractively, settable via the little triangles.
Menu:	Drop-down menu. Allows the user to choose one of several pre-set values.
URL:	Enter a URL in the field, click the button to show that URL in your browser.
App:	Enter some info (perhaps a filename), click the button, that application program starts and loads the entered info somehow (assumes that the application is predefined in the user-preferences file.

Because template requirements can change with usage, an additional tool was developed to allow template designers and end users to modify the style of a data element. The tool is fairly simple to use and is called Modify Properties. For example, the DC2S users often change text fields to graphics fields or vice versa as their understanding of the planning environment matures. Figure 11 shows an example of how a DC2S user can modify the properties of a data element. The *Modify Properties* tool includes a feature called *Attachment*. Using this feature, the user can attach a document to a field. The Find File option allows the user to navigate their machine for the desired attachment. Another feature allows the user to flag a field, indicating that the field is Essential. When this option is selected, the field will be delimited with an asterisk (*) specifying that the users must add a value.


Modify Properties [1]

Name: COMPONENT-ABBREV-FA

XPath: /CAP-Checklist/COMPONENT-ABBREV-FA

Pretty Name: Component Abbreviated FA

Field type: instance

URL/pathname

COMPONENT-ABBREV-FA.template

Edit Role:

Attachment:

Figure 11 - The Modify Properties Menu

5.7 Collaboration

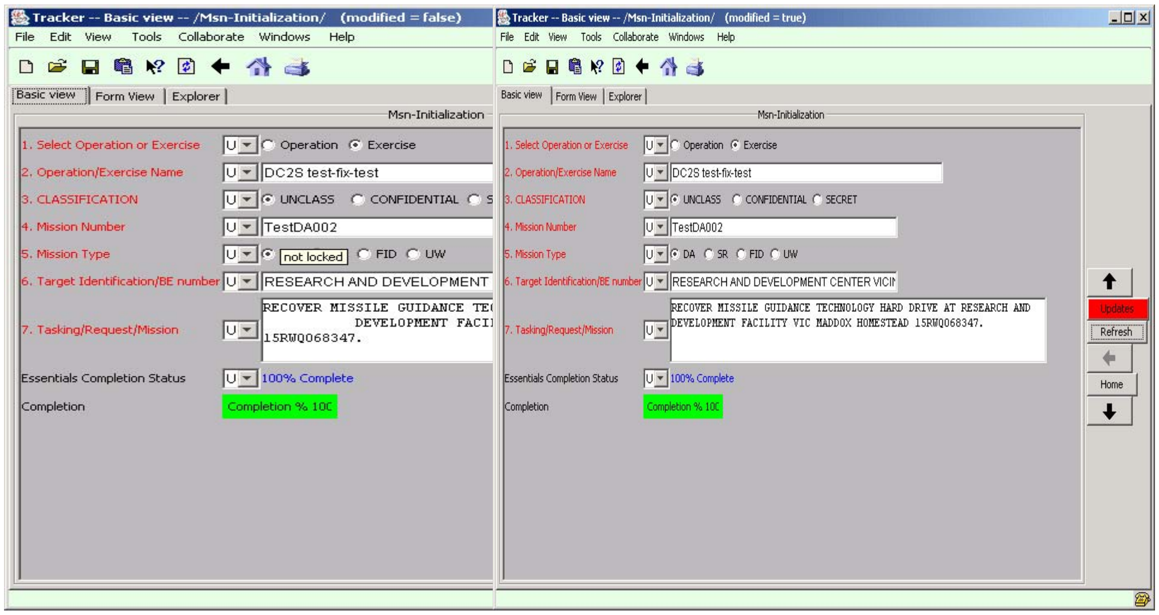
Special operations mission planning is a collaborative process. Information about the operational area is collected by a variety of individuals who function at different operational levels and in different operational roles. In order to support this type of plan development, collaborative tools were provided. Because we did not use a database to support collaboration, the prime method for collaboration required that multiple users connect to a common "host" user, open templates and build/save data.

In this protocol, one user must act as "Host", as with Microsoft NetMeeting [7]. Other users will "Join Meeting" with the Host. The host may specify a password if desired. Other users must enter the machine name or IP address, and the optional password. Following this, any user may open a template and then designate it as "shared".

When one of the collaboration participants designates a template as shared, it is transmitted to the server. Each user can then choose to open this template via menu commands (Collaborate>Open Shared Template), and select the template from the list of shared templates. All users who have the required edit-role for any particular data field may modify values for those fields. Note: The edit-roles were modeled on the actual command-and-control roles in the SOF planning environment. Locking is performed on a per-field basis, and is done immediately and automatically, and lock-release is done immediately after the field loses input focus. There is a potential race-condition in locking: if two users try to grab the same field in the same template at the same time, one will win and one will lose. This is network-speed and computer-speed dependent, but for any given collaboration group, like a Mission Planning Cell using DC2S, each player will have designated responsibilities which presumably do not interfere with others.

As a user makes data changes, those changes are communicated to the other users as follows. During collaboration, a change by another user will cause an update button (which appears on the right side in Basic View) to turn red. Clicking on this button will cause the display to refresh to show any changes. If the changes were made somewhere else in the template (in a sub-template that is not being displayed) the user will not see anything different, but the button will be red to indicate that there is a change somewhere in the template. (See Figure 12)

USER 1 **USER 2**



Tracker -- Basic view -- /Msn-Initialization/ (modified = false)

1. Select Operation or Exercise	U	Operation [✓ Exercise]
2. Operation/Exercise Name	U	DC2S test-fix-test
3. CLASSIFICATION	U	[✓ UNCLASS] CONFIDENTIAL S
4. Mission Number	U	TestDA002
5. Mission Type	U	● not locked FID UW
6. Target Identification/BE number	U	RESEARCH AND DEVELOPMENT
7. Tasking/Request/Mission	U	RECOVER MISSILE GUIDANCE TE DEVELOPMENT FACI 15RWQ068347.
Essentials Completion Status	U	100% Complete
Completion		Completion % 10C

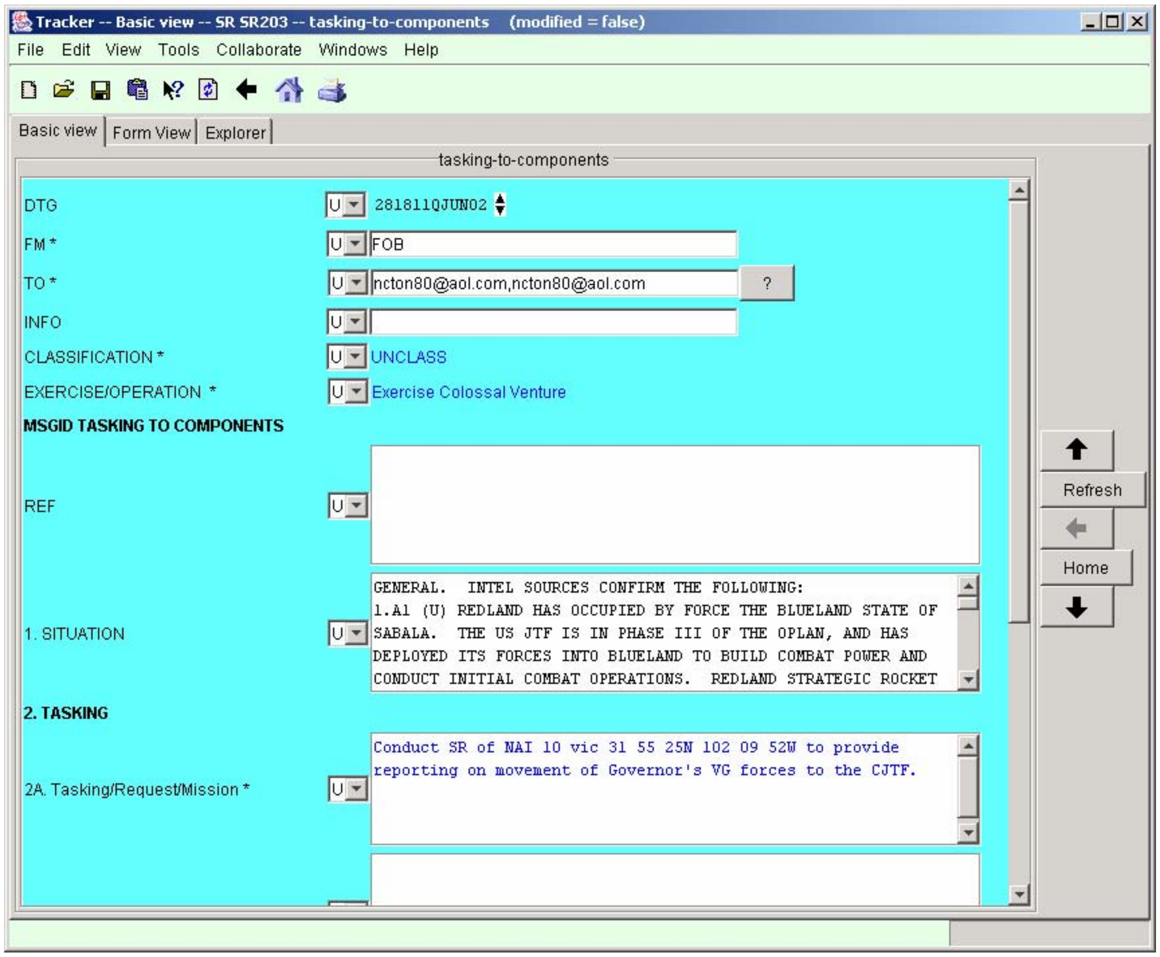
Tracker -- Basic view -- /Msn-Initialization/ (modified = true)

1. Select Operation or Exercise	U	Operation [✓ Exercise]
2. Operation/Exercise Name	U	DC2S test-fix-test
3. CLASSIFICATION	U	[✓ UNCLASS] CONFIDENTIAL SECRET
4. Mission Number	U	TestDA002
5. Mission Type	U	[✓ DA] SR FID UW
6. Target Identification/BE number	U	RESEARCH AND DEVELOPMENT CENTER VICIN
7. Tasking/Request/Mission	U	RECOVER MISSILE GUIDANCE TECHNOLOGY HARD DRIVE AT RESEARCH AND DEVELOPMENT FACILITY VIC MADDOX HOMESTEAD 15RWQ068347.
Essentials Completion Status	U	100% Complete
Completion		Completion % 10C

Figure 12- Collaboration and Updates

Any number of templates can be open and shared at a time. While there is undoubtedly a practical limit to this that is determined by the computer's operating system, *Tracker* doesn't have a built-in restriction. Note, during our DC2S experiments we found that the speed of the computers and network are the primary limiting factor.

5.7.1 Sending to Components

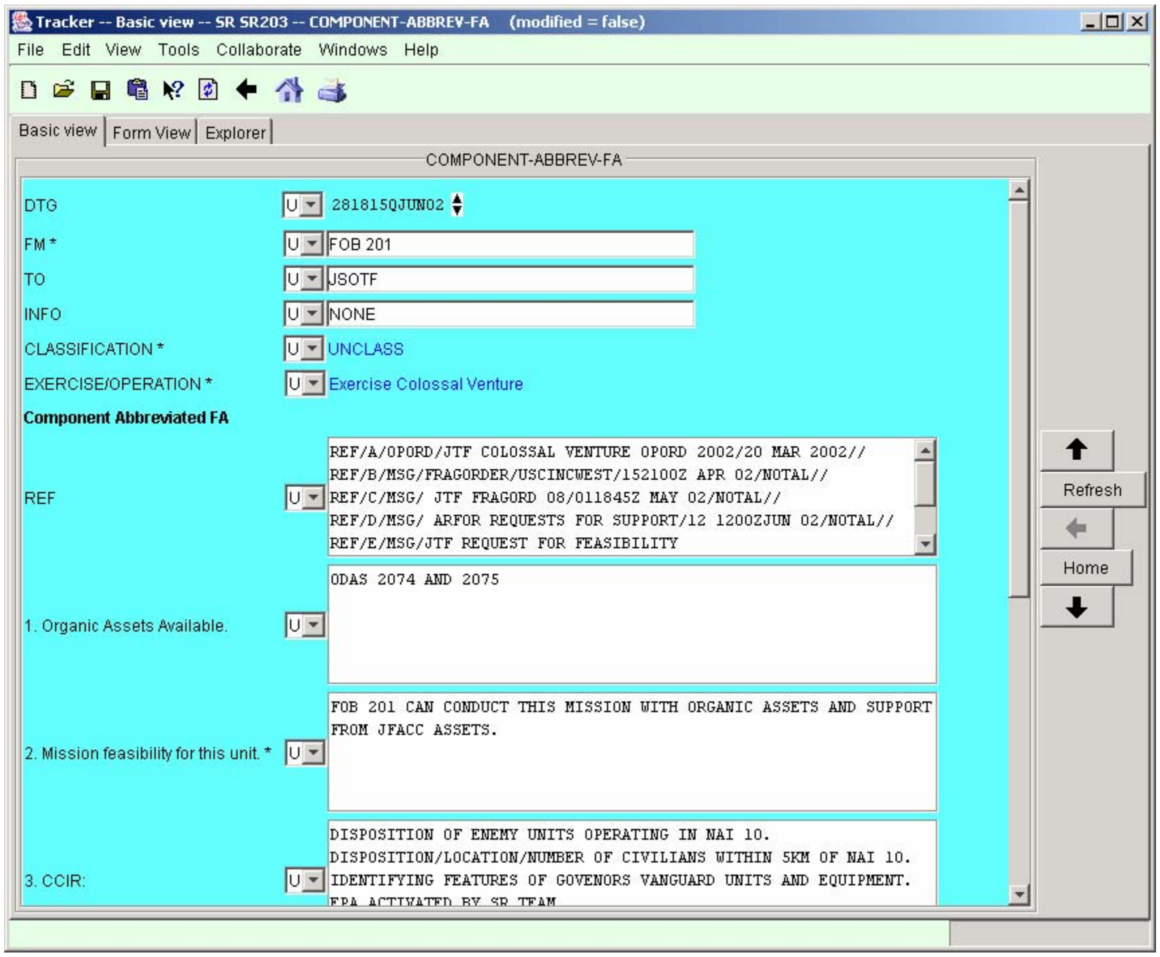

Tracker -- Basic view -- SR SR203 -- tasking-to-components (modified = false)
File Edit View Tools Collaborate Windows Help
Basic view
Form View
Explorer
tasking-to-components
Refresh
Home

DTG	U	281811QJUN02
FM *	U	FOB
TO *	U	ncton80@aol.com,ncton80@aol.com
INFO	U	
CLASSIFICATION *	U	UNCLASS
EXERCISE/OPERATION *	U	Exercise Colossal Venture
MSGID TASKING TO COMPONENTS		
REF	U	
1. SITUATION	U	GENERAL. INTEL SOURCES CONFIRM THE FOLLOWING: 1.A1 (U) REDLAND HAS OCCUPIED BY FORCE THE BLUELAND STATE OF SABALA. THE US JTF IS IN PHASE III OF THE OPLAN, AND HAS DEPLOYED ITS FORCES INTO BLUELAND TO BUILD COMBAT POWER AND CONDUCT INITIAL COMBAT OPERATIONS. REDLAND STRATEGIC ROCKET
2. TASKING		
2A. Tasking/Request/Mission *	U	Conduct SR of NAI 10 vic 31 55 25N 102 09 52W to provide reporting on movement of Governor's VG forces to the CJTF.

Figure 13 - Example of How Tasks are Sent to Components

One common form of collaboration in DC2S is based on the relationship between a commander and subcomponents. As a plan is being developed, the commander tasks subcomponents to perform certain tasks. In DC2S, the "To" field in the template "Tasking to Components" can be used to designate *whom* the subcomponents are (the values must look like email addresses, separated by commas). Figure 13 shows an example of two subcomponents being designated to receive the tasking via the "To" field in the template. Once the user saves this template, these two components will receive a set of templates via email. In addition, a folder will be created in the mission folder for the contents of their feasibility analysis input. If email is not available in a setting, the users can access their templates through the folders that have been created for the mission and exist on a shared drive or on a server.

The Component-Abbreviated-FA (Feasibility Analysis) is the main template that the components will complete and provide back to the sending organization. Figure 14 is an example of data as filled in by a subcomponent, e.g., FOB 201.



Tracker -- Basic view -- SR SR203 -- COMPONENT-ABBREV-FA (modified = false)

COMPONENT-ABBREV-FA

DTG	U	281815QJUN02
FM *	U	FOB 201
TO	U	JSOTF
INFO	U	NONE
CLASSIFICATION *	U	UNCLASS
EXERCISE/OPERATION *	U	Exercise Colossal Venture

Component Abbreviated FA

REF U

```
REF/A/OPORD/JTF COLOSSAL VENTURE OPORD 2002/20 MAR 2002//
REF/B/MSG/FRAGORDER/USCINCWEST/152100Z APR 02/NOTAL//
REF/C/MSG/ JTF FRAGORD 08/011845Z MAY 02/NOTAL//
REF/D/MSG/ ARFOR REQUESTS FOR SUPPORT/12 1200ZJUN 02/NOTAL//
REF/E/MSG/JTF REQUEST FOR FEASIBILITY
```

1. Organic Assets Available. U

```
ODAS 2074 AND 2075
```

2. Mission feasibility for this unit. * U

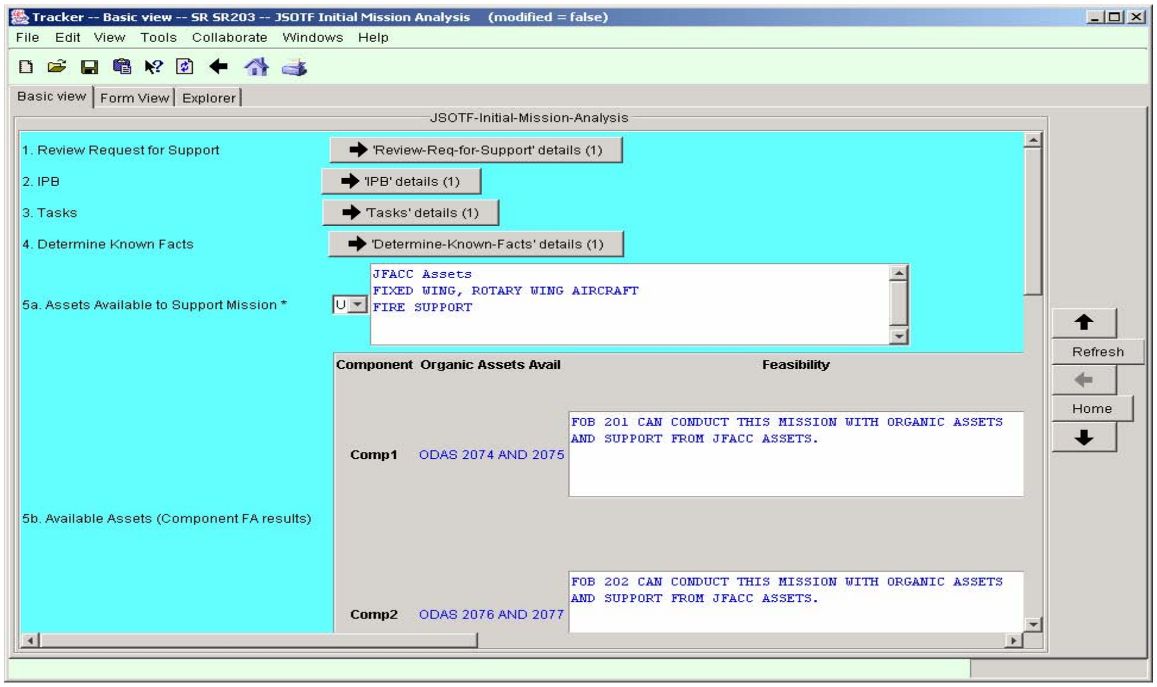
```
FOB 201 CAN CONDUCT THIS MISSION WITH ORGANIC ASSETS AND SUPPORT
FROM JFACC ASSETS.
```

3. CCIR: U

```
DISPOSITION OF ENEMY UNITS OPERATING IN NAI 10.
DISPOSITION/LOCATION/NUMBER OF CIVILIANS WITHIN 5KM OF NAI 10.
IDENTIFYING FEATURES OF GOVENORS VANGUARD UNITS AND EQUIPMENT.
EPA ACTIVATED BY SR TEAM
```

Figure 14 - Example Component-Abbreviated-Feasibility-Analysis Template

With the data received from each Component, the higher commander, e.g., Joint Special Operations Task Force (JSTOF) will use it to build the “Initial Mission Analysis”. Figure 15 presents a view of the data from two components that were tasked in Figure 13. The data values in blue are values that were automatically derived from the templates that were filled in by the subcomponents. Note: The general protocol is that values in DC2S that are colored blue mean that they have been derived (slaved) from other sources.


Tracker -- Basic view -- SR SR203 -- JSOTF Initial Mission Analysis (modified = false)
File Edit View Tools Collaborate Windows Help
Basic view
Form View
Explorer
Refresh
Home

JSOTF-Initial-Mission-Analysis

1. Review Request for Support — 'Review-Req-for-Support' details (1)

2. IPB — 'IPB' details (1)

3. Tasks — 'Tasks' details (1)

4. Determine Known Facts — 'Determine-Known-Facts' details (1)

5a. Assets Available to Support Mission * — U

```
JFACC Assets
FIXED WING, ROTARY WING AIRCRAFT
FIRE SUPPORT
```

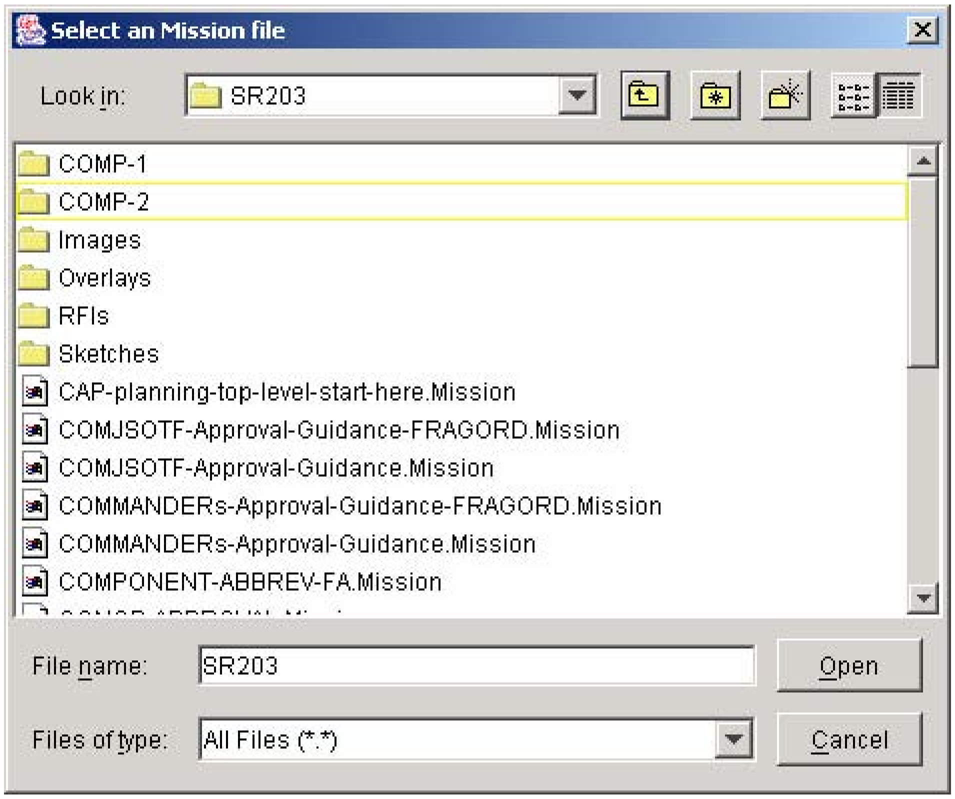
5b. Available Assets (Component FA results)

Component	Organic Assets Avail	Feasibility
Comp1	ODAS 2074 AND 2075	FOB 201 CAN CONDUCT THIS MISSION WITH ORGANIC ASSETS AND SUPPORT FROM JFACC ASSETS.
Comp2	ODAS 2076 AND 2077	FOB 202 CAN CONDUCT THIS MISSION WITH ORGANIC ASSETS AND SUPPORT FROM JFACC ASSETS.

Figure 15 - Example JSOTF-Initial-Mission-Analysis Template

As the components and the JSOTF collect their mission data, they can store the data in the folders that are associated with the mission folder that they create. Note: This mission folder was automatically created as a side effect of instantiating a new SOF mission. Figure 16 displays an example of a mission folder. The data from the two subcomponents tasked in Figure 13 are stored in COMP-1 and COMP-2 folders. Basically, a COMP folder is created for each subcomponent addressed in the "To" field of the Tasking-to-components template.

Any images that the user would like to use in this mission folder should be copied or stored in the Images folder. This is likewise true for Overlays, RFI (Request for Information), and Sketches folders.


Select an Mission file

Look in: SR203

- COMP-1
- COMP-2
- Images
- Overlays
- RFIs
- Sketches
- CAP-planning-top-level-start-here.Mission
- COMJSOTF-Approval-Guidance-FRAGORD.Mission
- COMJSOTF-Approval-Guidance.Mission
- COMMANDERs-Approval-Guidance-FRAGORD.Mission
- COMMANDERs-Approval-Guidance.Mission
- COMPONENT-ABBREV-FA.Mission

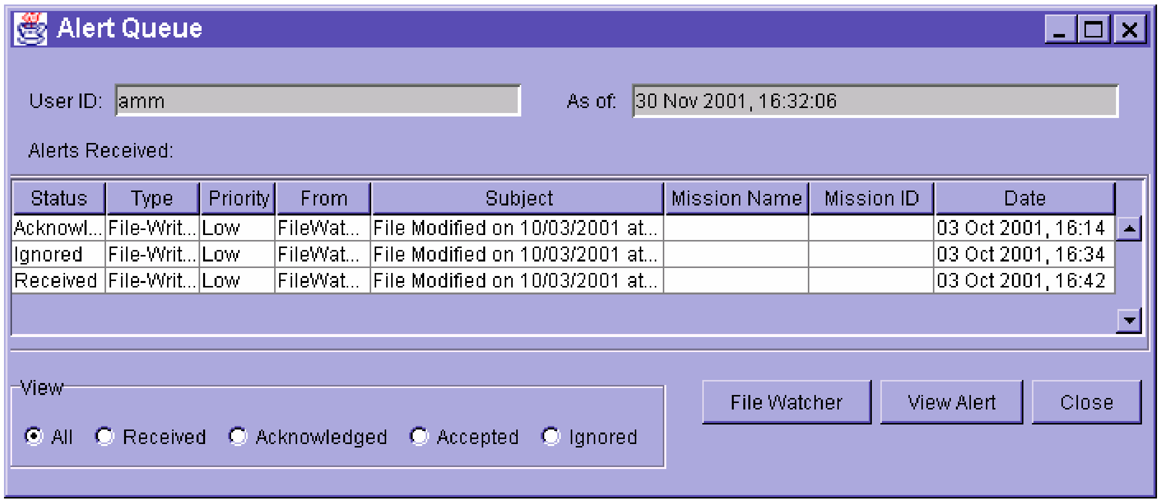
File name: SR203

Files of type: All Files (*.*)

Figure 16 - Example Mission Folder Structure; Note Images Folder for Storing Images

5.8 Generating Alerts

In most circumstances, a plan will change both prior to and during execution. Since planners typically want to be made aware of changes, we developed a tool to support the identification and generation of alerts. Figure 17 is an example of the alert mechanism that was developed.


Alert Queue

User ID: amm

As of: 30 Nov 2001, 16:32:06

Alerts Received:

Status	Type	Priority	From	Subject	Mission Name	Mission ID	Date
Acknowl...	File-Writ...	Low	FileWat...	File Modified on 10/03/2001 at...			03 Oct 2001, 16:14
Ignored	File-Writ...	Low	FileWat...	File Modified on 10/03/2001 at...			03 Oct 2001, 16:34
Received	File-Writ...	Low	FileWat...	File Modified on 10/03/2001 at...			03 Oct 2001, 16:42

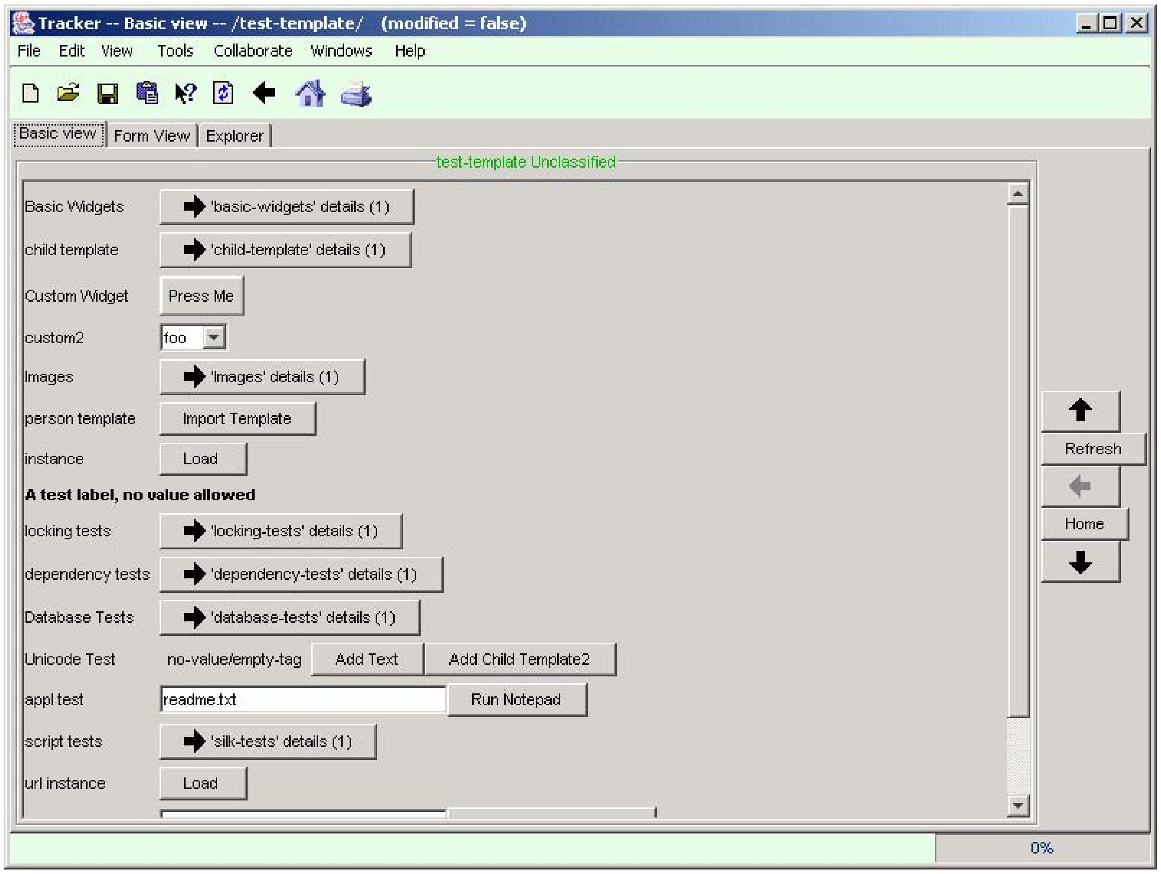
View: [✓ All] Received Acknowledged Accepted Ignored

Figure 17 - Example of the Alert Queue

Unfortunately, the DC2S users did not find this alert queue concept to be of use for them. Instead, they preferred to rely both on the red highlights that were generated to correspond to some change during collaborative planning, and on the percent-complete value available in the templates that indicates when required fields were specified.

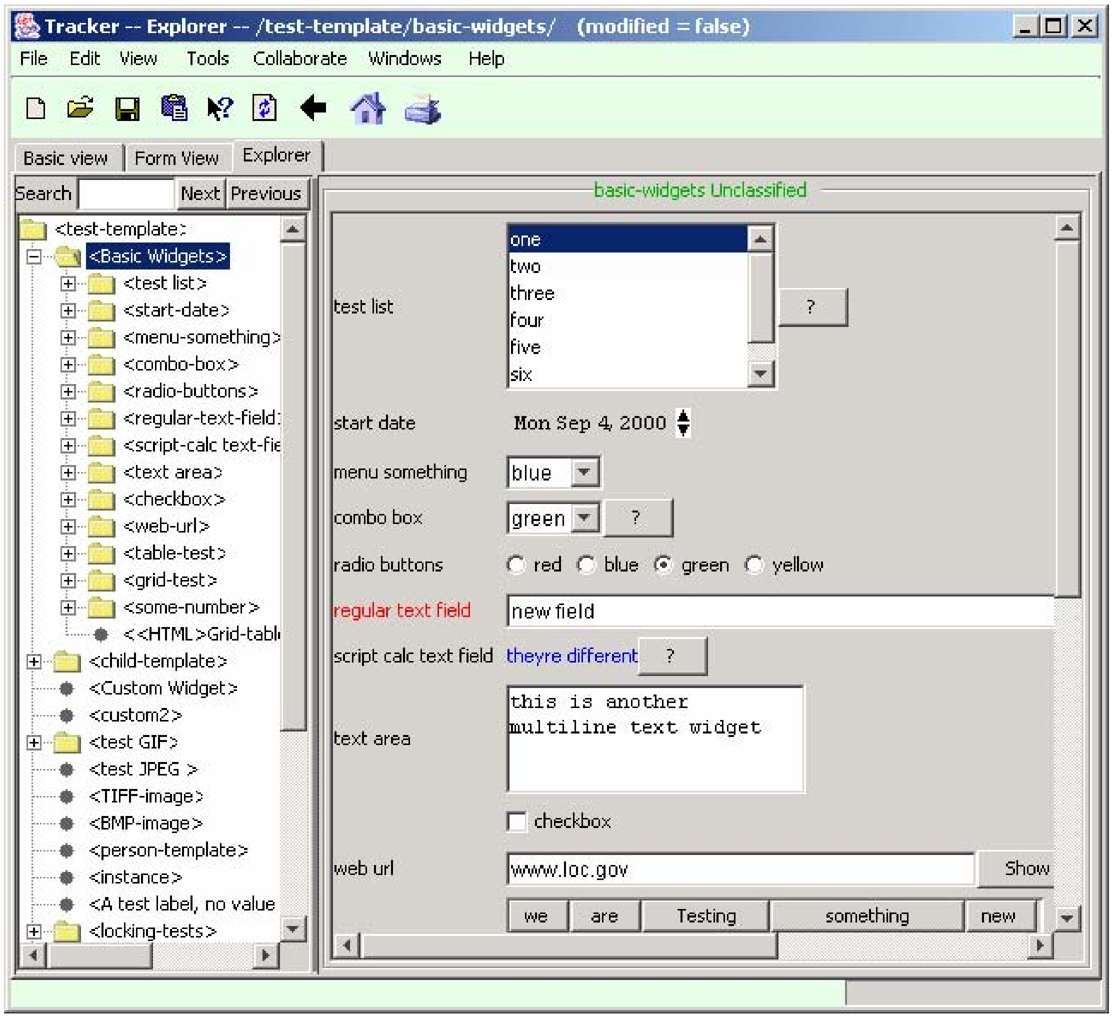
5.9 Using Scripts and Other Widgets

Although slaving (the sharing of data from a slot to a related slot) is a powerful way to propagate data values, scripts are used in DC2S (and in the more general *Tracker*) to support database queries and more complicated computations. Scripts are based on the programming language JScheme [8]. Various attempts were made through the lifetime of the project to make the creation of these scripts easier for a common user. The most popular technique was to allow the user to find a script that worked in a way similar to what the user wanted to do, and to simply modify the arguments to suit a new problem domain. In order to support this activity, several examples of scripts are provided in the "test template" and "basic widgets" template (see Figure 18 and 19) that are provided with *Tracker* and DC2S to support authoring and script generation.


Tracker -- Basic view -- /test-template/ (modified = false)
File Edit View Tools Collaborate Windows Help
Basic view
Form View
Explorer
test-template Unclassified
Refresh
Home
0%

Basic Widgets 'basic-widgets' details (1)
child template 'child-template' details (1)
Custom Widget Press Me
custom2 foo
Images 'Images' details (1)
person template Import Template
instance Load
A test label, no value allowed
locking tests 'locking-tests' details (1)
dependency tests 'dependency-tests' details (1)
Database Tests 'database-tests' details (1)
Unicode Test no-value/empty-tag Add Text Add Child Template2
appl test readme.txt Run Notepad
script tests 'silk-tests' details (1)
url instance Load

Figure 18 - Basic Tracker Test Template



test list: one, two, three, four, five, six

start date: Mon Sep 4, 2000

menu something: blue

combo box: green

radio buttons: red, blue, [✓ green], yellow

regular text field: new field

script calc text field: theyre different

text area: this is another multiline text widget

checkbox

web url: www.loc.gov

Figure 19 - Basic Tracker Computational Widgets

5.10 DC2S Summary

ARSOBL conducted several evaluations of DC2S and nominated it for use and testing during the Millennium Challenge 2002 (MC02) Exercise. The MC02 experience was valuable because it indicated that although DC2S was very powerful, it required a bit more training than was desired by the end users. We also experienced problems with the collaboration tool. It was slow at times and values got locked because all of the users decided to have “root” (write controlling) access. In the end, ARSOBL opted to continue their work with another tool that was being developed for another SOF component by other researchers in the AcT program. This tool had less of a training curve. In a test conducted after the choice, DC2S could do all and more than the tool that they chose. The lesson learned was that “less is sometimes better.”

6. Other Research Experiments

6.1 Workflow Development

Tracker templates can be built “on the fly” or with the support of specific data models (like the workflow diagram displayed in Figure 20 that describe crisis action planning for a fire incident). Our experience with template development indicates that a data model based on a combination of two input factors: (a) doctrine or protocol, and (b) domain scenarios; facilitates the authoring of templates and increases the likelihood that the resulting system will meet the needs of the end user. A scenario, when provided by a user, can be used to identify what templates need to be built, how the templates should interact, and what external data sources should be accessed in order to support information usage. A scenario can also be used to support experimentation and testing, and it provides an environment by which to evaluate how a set of users might collaborate with the templates.

Several small experiments were conducted to build workflow engines to support a variety of domains. Figure 21 shows one of the workflow templates built in *Tracker* to support the Fire Department workflow that is pictured in Figure 20. Other workflow templates were developed to support other domains, e.g., travel planning.

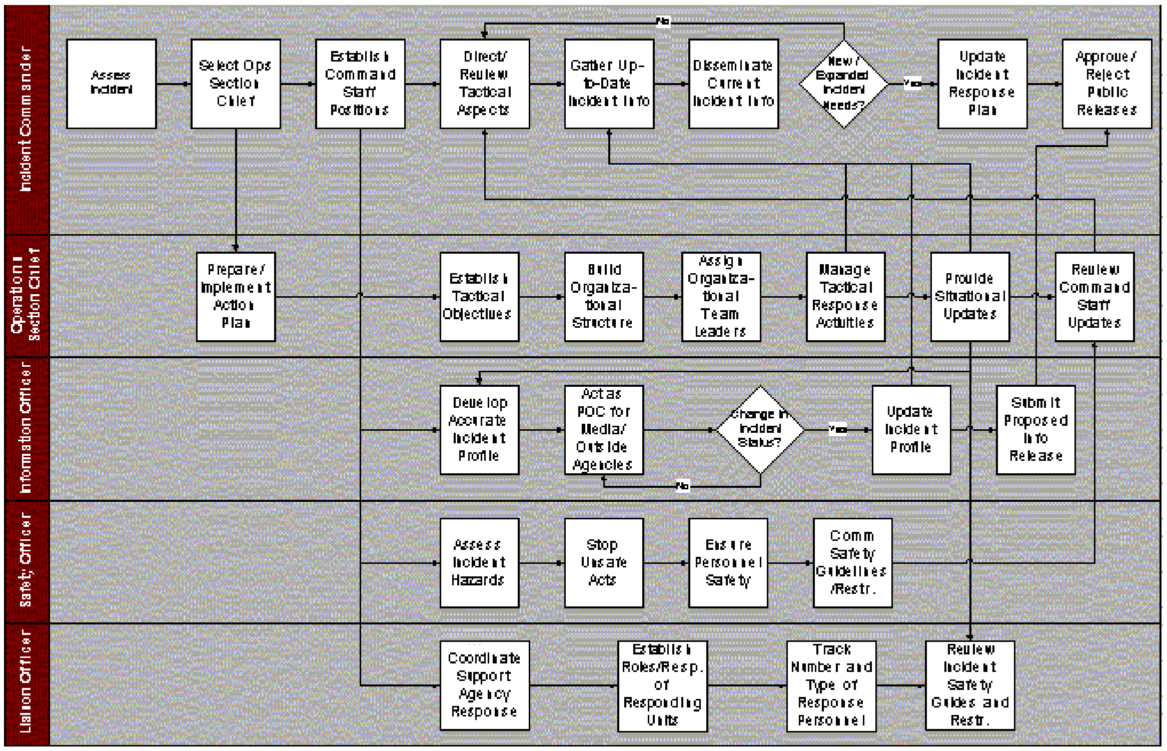
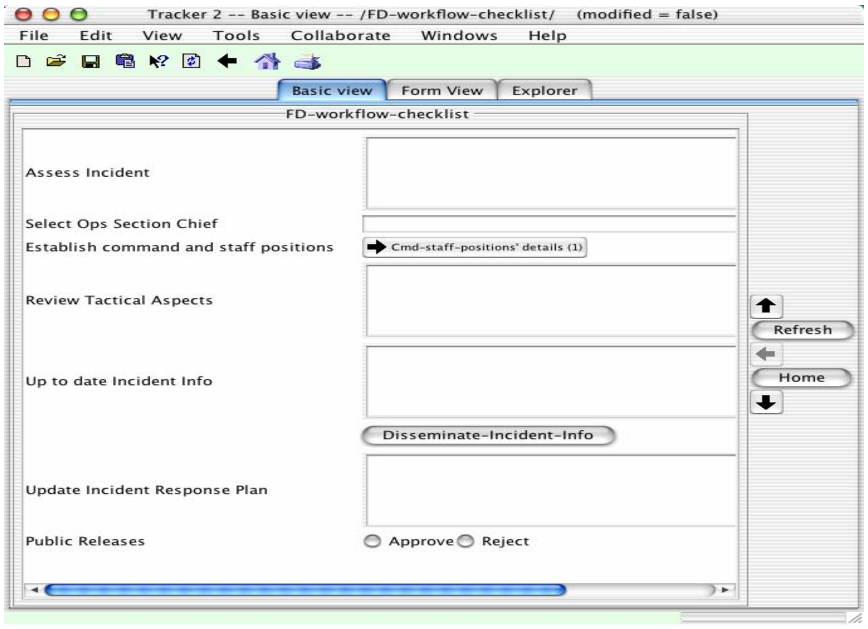

Incident Commander
Operations Section Chief
Information Officer
Safety Officer
Liaison Officer
Assess Incident
Select Ops Section Chief
Establish Command Staff Positions
Direct/ Review Tactical Aspects
Gather Up-to-Date Incident Info
Disseminate Current Incident Info
New / Expanded Incident Needs?
No
Yes
Update Incident Response Plan
Approve / Reject Public Releases
Prepare / Implement Action Plan
Establish Tactical Objectives
Build Organizational Structure
Assign Organizational Team Leaders
Manage Tactical Response Activities
Provide Situational Updates
Review Command Staff Updates
Develop Accurate Incident Profile
Act as POC for Media/ Outside Agencies
Change in Incident Status?
Yes
No
Update Incident Profile
Submit Proposed Info Release
Assess Incident Hazards
Stop Unsafe Acts
Ensure Personnel Safety
Comm Safety Guidelines /Restr.
Coordinate Support Agency Response
Establish Roles/Resp. of Responding Units
Track Number and Type of Response Personnel
Review Incident Safety Guides and Restr.

Figure 20 - An Example Workflow with Information Requirements for a Fire Department Plan



Tracker 2 -- Basic view -- /FD-workflow-checklist/ (modified = false)

FD-workflow-checklist

Assess Incident

Select Ops Section Chief

Establish command and staff positions — Cmd-staff-positions' details (1)

Review Tactical Aspects

Up to date Incident Info

Disseminate-Incident-Info

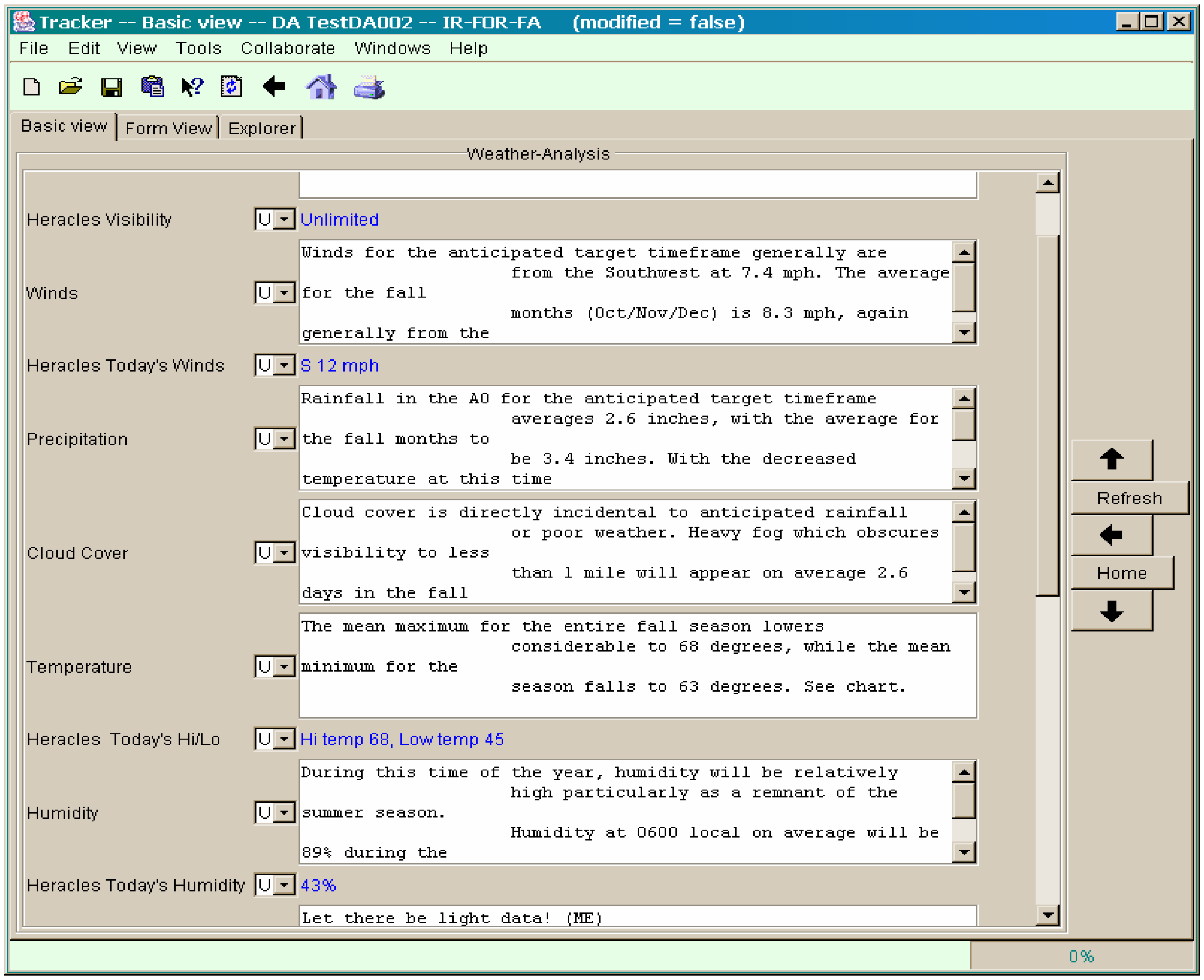
Update Incident Response Plan

Public Releases — ○ Approve ○ Reject

Figure 21 - The Tracker Version of the Fire Department Workflow

6.2 Real Time Data Collection and Updates

We conducted an experiment with an online service provided by Heracles (another AcT tool that was developed by researchers at the Information Sciences Institute [9]). During this experiment, we built an interface to Heracles to collect real-time data that was available through the tool. We were able to build special fields that called Heracles to obtain information that augmented data collected in DC2S templates. For example, in Figure 22, you can see some standard textual fields used to support the description of weather in a DC2S template. You will also see several additional fields, e.g., Heracles Today's Winds, Heracles Visibility, and Heracles Today's Hi/Lo, that derive their values through a call to the Heracles tool.

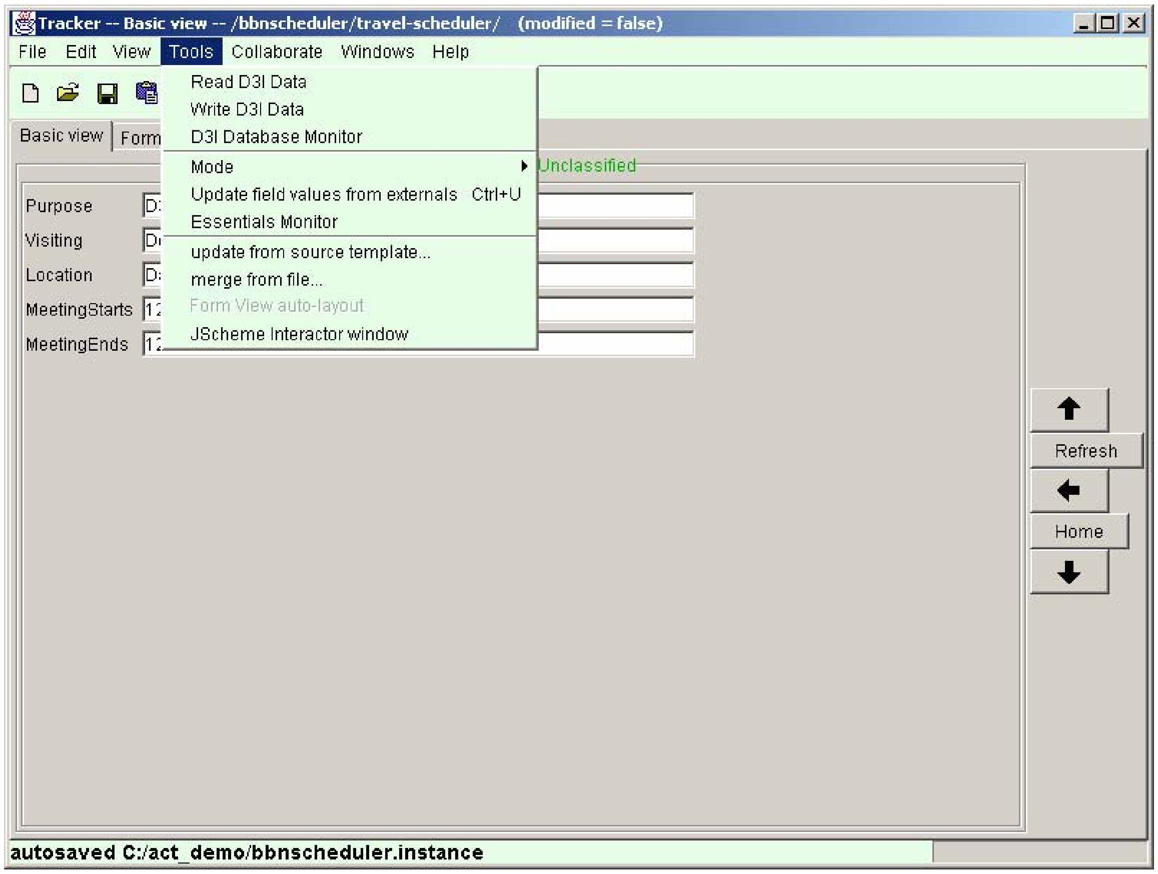

Tracker -- Basic view -- DA TestDA002 -- IR-FOR-FA (modified = false)
File Edit View Tools Collaborate Windows Help
Basic view
Form View
Explorer
Weather-Analysis
Refresh
Home
0%

Heracles Visibility	U	Unlimited
Winds	U	Winds for the anticipated target timeframe generally are from the Southwest at 7.4 mph. The average for the fall months (Oct/Nov/Dec) is 8.3 mph, again generally from the
Heracles Today's Winds	U	S 12 mph
Precipitation	U	Rainfall in the AO for the anticipated target timeframe averages 2.6 inches, with the average for the fall months to be 3.4 inches. With the decreased temperature at this time
Cloud Cover	U	Cloud cover is directly incidental to anticipated rainfall or poor weather. Heavy fog which obscures visibility to less than 1 mile will appear on average 2.6 days in the fall
Temperature	U	The mean maximum for the entire fall season lowers considerable to 68 degrees, while the mean minimum for the season falls to 63 degrees. See chart.
Heracles Today's Hi/Lo	U	Hi temp 68, Low temp 45
Humidity	U	During this time of the year, humidity will be relatively high particularly as a remnant of the summer season. Humidity at 0600 local on average will be 89% during the
Heracles Today's Humidity	U	43%
		Let there be light data! (ME)

Figure 22 - Heracles Weather Data Pulled From a Tracker Template

6.3 Experiment with D3i, a Tool Developed by DARPA

D3i [10] is another Active Templates tool that was developed as part of the DARPA AcT program. The API for communication between D3i and *Tracker* is displayed in Figure 23.


Tracker -- Basic view -- /bbnscheduler/travel-scheduler/ (modified = false)

autosaved C:/act_demo/bbnscheduler.instance

Figure 23 - D3i API

The idea behind D3i is that the user should be able to find and re-use templates or template fragments from a variety of AcT template tools. D3i uses a database to store templates and instances. During our experiments with D3i, we extended our database capabilities to read and write to the D3i database and to share both instance and template data. This work also propelled us to further develop the “browser” tool in order to focus on better ways to perform searches of both structured and unstructured data. Several members of the team investigated related research topics in this area, including issues related to mining XML data, and related to various browsing techniques that are currently available for large amounts of XML data. One method involves XForms [11], a W3C specification that is part of Office 2003, and evidently the basis for MS InfoPath [12], a Microsoft capability that mirrors a lot of the Active Templates research ideas/capabilities. The result of these investigations was the development of a more powerful browser that ran independent of *Tracker*. This independent Browser could interact with *Tracker* or other Active Template applications (e.g., D3i and CommandLink) through a copy/paste method that allowed a user to find templates and copy them from one AcT application into another.

6.4 CommandLink

CommandLink is another of the AcT tools that we conducted experiments with. It was developed to support a different set of SOF users, and it has many of the features that *Tracker* has. BBN conducted a comparison between CommandLink version 1.4 and *Tracker* version 0.74 during this project. Over time, efforts were made to interchange templates with

CommandLink, and as a result of the comparison, many enhancements were made to both tools. The following table describes the differences and similarities between these two tools.

Table 2- CommandLink - Tracker Comparison

	TRACKER	COMMANDLINK
Create New Form	YES	YES
Open Existing Form	YES	YES
Save Form	YES	Done automatically
Entry/Text Field	YES	YES
NumberEntry/Integer	YES	YES
RadioButtonSet	YES	YES
ComboBox/Menu	YES	YES
Text Area	YES	YES
Scale/Slider	YES	YES
CheckButton/CheckBox	YES	YES
CheckButtonSet	???	YES
StopLight	NO	YES
StopLightSet	NO	YES
Field for Details (Childern)	YES	NONE – There is no notion of a hierarchy of forms
Label	YES	Labels are added when attributes are added – cannot be added separately
Date Field	YES	NO
URL Field	YES	NO
Database Link Field	YES	NO
Table Field	YES	NO
Application	YES	NO
Action	YES	NO
Custom Widget	YES	NO
Image	YES	NO (to be added in version 2)
Instance	YES	NO
Remote XPATH	YES	NO (to be added in version 2)
Remote DB	YES	NO (to be added in version 2)
Silk Script	YES	NO
Slave/Linked Source	YES	YES
Table	YES	NO

	TRACKER	COMMANDLINK
Template	YES	NO
Save Form as…	YES	NO
Save Form as Pattern	NO	YES
Sharing of Forms	YES	YES
Email Form	YES	YES but only to people connected to the router at that time
Edit User Preferences	YES	Yes but limited
Print Form	YES	NO
Close Form	YES	NO Just “X”
Quit	YES	NO Just “X”
Show User Action History	YES	NO

6.5 DAML

Ontologies are a descriptive approach for specifying relationships between things (information). Relationships are often not hierarchical, so an ontology defines a graph rather than a tree. The DARPA Agent Markup Language (DAML) program was established to develop tools that could create and utilize ontologies [13]. During the course of the DAML program, the Ontology Web Language (OWL) emerged [14]. Both DAML and OWL are based on XML. We conducted several experiments to use *Tracker* to read, write, and generate DAML/OWL data files. *Tracker* was modified to follow RDF links in addition to *Tracker*’s own template links. As a result, *Tracker* can open and display a DAML/OWL/RDF file, and the internal links can be navigated in just the right way (even when that link points to another DAML file on a web-server—a click on the link button opens that other DAML file). However, the graph structure of DAML/OWL does not typically lend itself to clean on-screen presentation in *Tracker*’s usual style, so a DAML/OWL graph can be very difficult to understand when viewed in *Tracker*. We realized that a better approach would be to create a custom view for DAML/RDF data, or a custom widget, but we really had no need to do so because there was no application requiring it. Instead we developed a tool to export *Tracker* templates in a number of XML styles (using stylesheets), and these exported files have been successfully used as input to DAML/OWL based systems, such as the ACT Foreign Clearance System (described below).

6.6 Web Planner

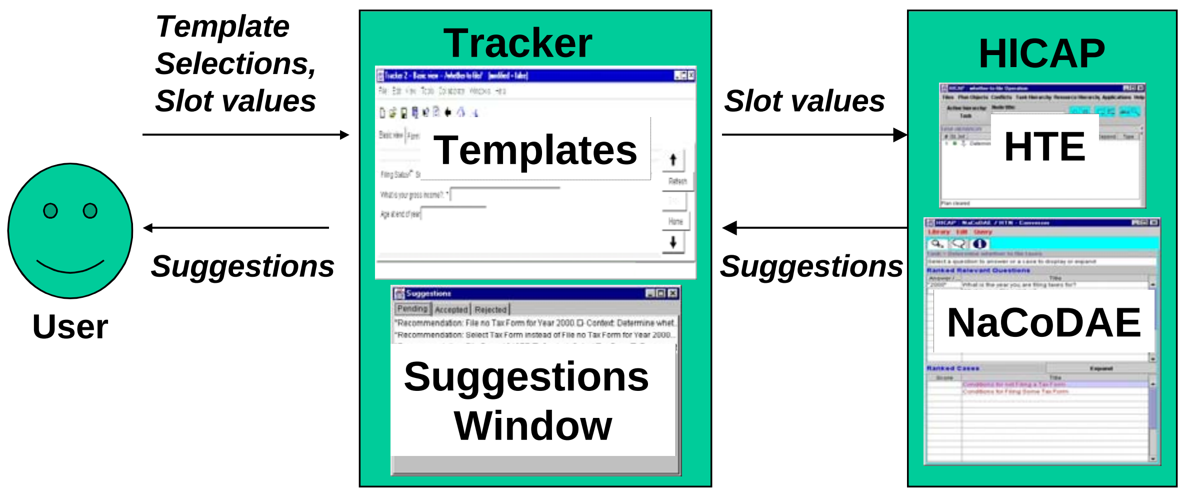
The DARPA Adaptive Course of Action (ACOA) program’s Web Planner is a tool that creates instances of planning document structures, which are very much like *Tracker* templates. At one point as an experiment we duplicated a portion of this structure as *Tracker* templates. This work was never finished. It was done as an experiment to determine how much effort it takes to create (or duplicate) structures like those in Web Planner with *Tracker*. Note: Web Planner’s “templates” are similar in structure to *Tracker* templates, with nested substructure, pick-lists, text fields, etc.; hence, constructing Web Planner templates in *Tracker* took little time.

6.7 HiCAP

HiCAP is another of the AcT tools. It is an interactive case-based tool that supports planning. HiCAPs planning module includes a case-base and a generative planner. The generative planner in HiCAP is provided by the planner SHOP [15], while the case-based planner is based on the Navy Conversational Decision Aids Environment (NaCoDAE) [16]. We conducted an experiment to use the planning tools of HiCAP to provide suggestions on how a user should select and fill out a set of templates. For this experiment, we began with the IRS tax-forms as a domain model. Figure 24 shows the concept proposed for communication between *Tracker* and HiCAP. Figure 25 shows Tax Templates provided by *Tracker,* and suggestions provided by HiCAP.

Our goal in this experiment was to demonstrate how suggestions provided to the DC2S users could enhance the usage of the tool. While we were able to demonstrate that *Tracker* could work with a planner to better support an end user, we were not able to encourage the ARSOBL SOF users to support this extension.

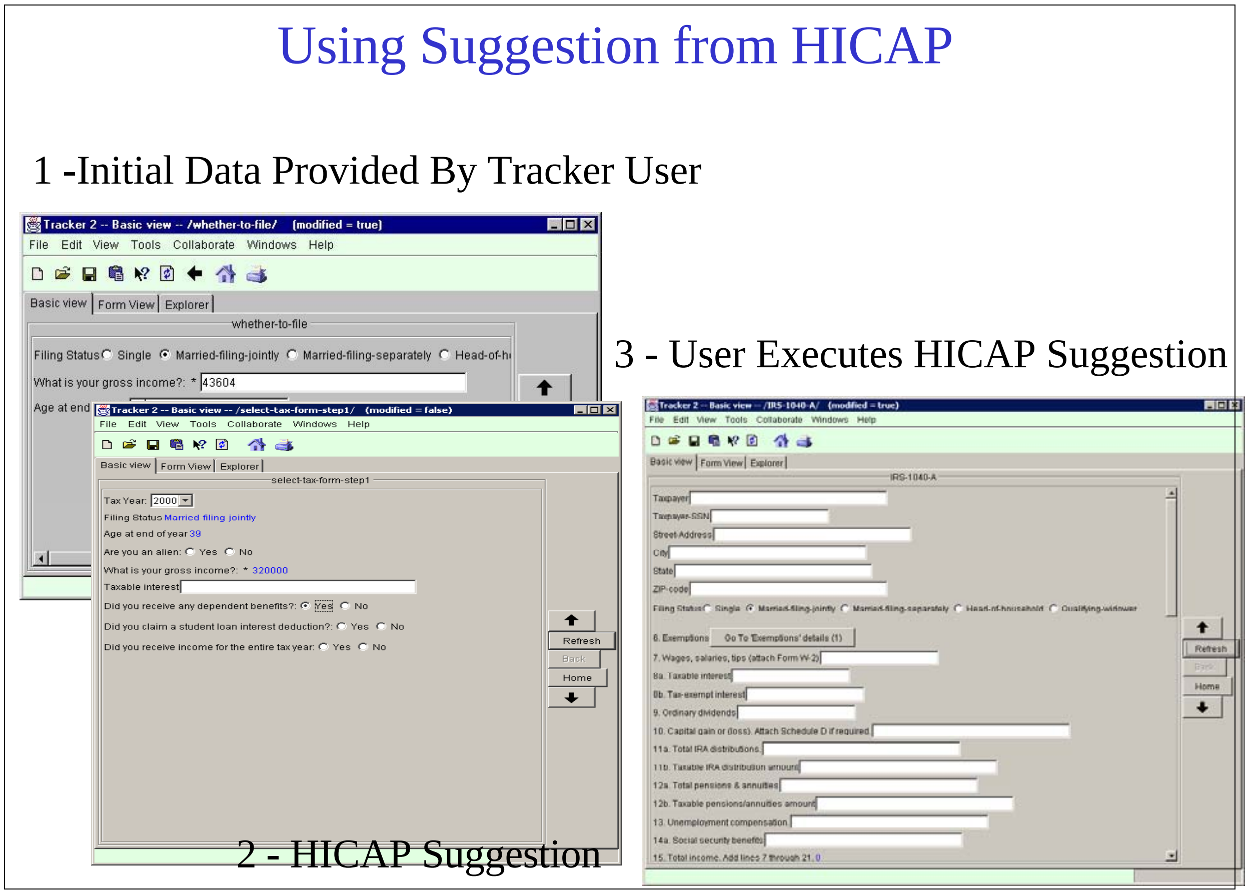
Initial Tracker/HICAP Interactions



Suggestion types:

1. Open a <specific> template
2. Switch to another <specified> template
3. Close a <specific> template
4. Insert a <specified> value for a <specific> slot

Figure 24 - The Tracker/HICAP Concept Overview


Using Suggestion from HICAP
1 -Initial Data Provided By Tracker User
3 - User Executes HICAP Suggestion
2 - HICAP Suggestion
File Edit View Tools Collaborate Windows Help
Basic view | Form View | Explorer
Refresh
Back
Home

Tracker 2 -- Basic view -- /whether-to-file/ (modified = true)

whether-to-file

Filing Status ○ Single [✓ Married-filing-jointly] ○ Married-filing-separately ○ Head-of-h

What is your gross income?: * 43604

Age at end

Tracker 2 -- Basic view -- /select-tax-form-step1/ (modified = false)

select-tax-form-step1

Tax Year: 2000

Filing Status Married-filing-jointly

Age at end of year 39

Are you an alien: ○ Yes ○ No

What is your gross income?: * 320000

Taxable interest

Did you receive any dependent benefits?: [✓ Yes] ○ No

Did you claim a student loan interest deduction?: ○ Yes ○ No

Did you receive income for the entire tax year: ○ Yes ○ No

Tracker 2 -- Basic view -- /IRS-1040-A/ (modified = true)

IRS-1040-A

Taxpayer

Taxpayer-SSN

Street-Address

City

State

ZIP-code

Filing Status ○ Single [✓ Married-filing-jointly] ○ Married-filing-separately ○ Head-of-household ○ Qualifying-widower

6. Exemptions Go To 'Exemptions' details (1)

7. Wages, salaries, tips (attach Form W-2)

8a. Taxable interest

8b. Tax-exempt interest

9. Ordinary dividends

10. Capital gain or (loss). Attach Schedule D if required

11a. Total IRA distributions

11b. Taxable IRA distribution amount

12a. Total pensions & annuities

12b. Taxable pensions/annuities amount

13. Unemployment compensation

14a. Social security benefits

15. Total income. Add lines 7 through 21. 0

Figure 25 - Tracker Tax Templates with HICAP suggestions

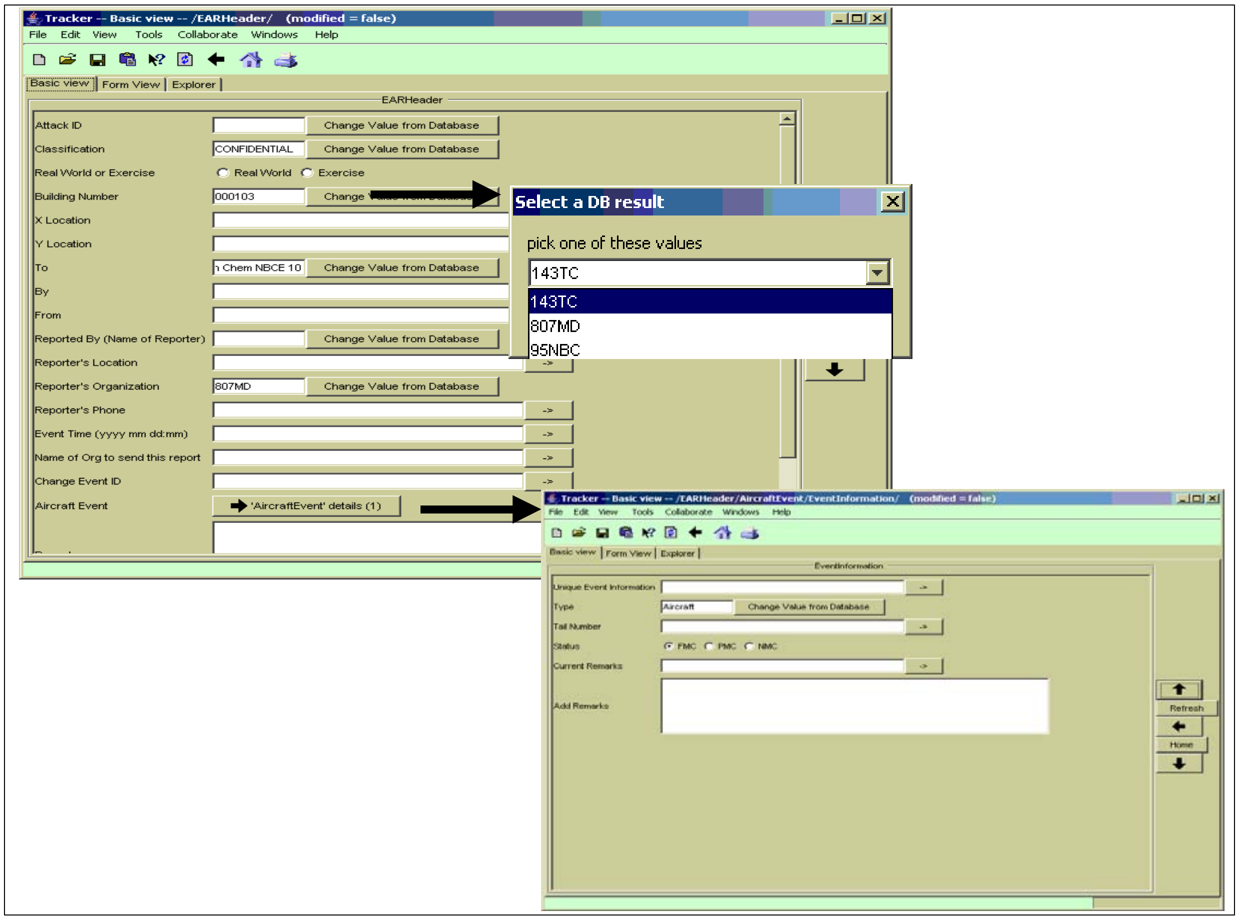
Two demonstrations of this work were made. One of the demonstration scripts is provided in Appendix A, along with the API that was used to facilitate communication between HiCAP and *Tracker*.

6.8 IWARN, An AFRL Incident Reporting Environment

The Installation, Warning, and Reporting Network (IWARN) is an AFRL incident reporting environment. Developers of IWARN's Digital Dashboard tool expressed an interest in using *Tracker* to support the development of templates for incident collection. IWARN is a suite of information technologies that fold chemical and biological sensors, networks, command and control, situation awareness, Nuclear-Biological-Chemical (NBC) modeling and analysis, and data fusion into a seamless information and decision-support system. The technology underlying IWARN provides the means for fusing information from the many passive, active, and human data sources that are associated with the detection and tracking of chemical and biological attacks, both overt and covert.

In order to provide decision makers with timely situational awareness, IWARN leverages the concept of a *Digital Dashboard*. *Digital Dashboards* are often used to gather information about a problem and provide a set of visualization methods that describe the state of a problem domain at a given time. In IWARN, a variety of Electronic Attach Report (EAR) forms are provided to support data collection. IWARN uses a database for storage of EAR data.

In order to support the usage of *Tracker* developed templates by IWARN, we developed several new template widgets to support database access (See Figure 23). We also investigated integration approaches, in particular the use of *Tracker* as a cell in the IWARN dashboard and the use of *Tracker* as a callable product. Our research indicated that although the tighter integration achieved by using *Tracker* as a cell was most desirable by the IWARN researchers, Tracker was developed to run as a standalone callable product. Work beyond the scope of the experiment would have been required to achieve greater integration.


File Edit View Tools Collaborate Windows Help
Basic view | Form View | Explorer
Change Value from Database
->
Refresh
Home

Tracker -- Basic view -- /EARHeader/ (modified = false)

EARHeader

Attack ID	
Classification	CONFIDENTIAL
Real World or Exercise	Real World / Exercise
Building Number	000103
X Location	
Y Location	
To	n Chem NBCE 10
By	
From	
Reported By (Name of Reporter)	
Reporter's Location	
Reporter's Organization	807MD
Reporter's Phone	
Event Time (yyyy mm dd:mm)	
Name of Org to send this report	
Change Event ID	
Aircraft Event	'AircraftEvent' details (1)

Select a DB result

pick one of these values

143TC

143TC
807MD
95NBC

Tracker -- Basic view -- /EARHeader/AircraftEvent/EventInformation/ (modified = false)

EventInformation

Unique Event Information	
Type	Aircraft
Tail Number	
Status	[✓ FMC] PMC NMC
Current Remarks	
Add Remarks	

Figure 26 - EAR Templates as Developed in Tracker

During our experiment with IWARN, we were able to replicate several of the EAR forms, and we discovered that by providing the specific usage of database tables we could communicate with the IWARN database without corrupting it with modified templates. To handle template modifications and extensions we developed a method to store additional fields in a table. This table would then be provided to the IWARN database administrator, who would then determine when frequently added data fields should result in IWARN database schema updates.

We also discovered that *Tracker* templates could be integrated more easily into other software if the authoring widgets were separated from the core *Tracker* code. To allow the proper sharing of widgets, a widget library with a common API would need to be defined. This API would allow the development of multiple platform specific libraries without the need to modify the core of *Tracker*. Note: any future extensions to Tracker would probably include the development of this API.

6.9 Automated Clearance Tool (ACT)

The Automated Clearance Tool (ACT) is an agent based tool that uses semantically annotated data sources to compute the lead time required for AMC to obtain diplomatic clearances for their missions [17]. Most of the data entry/processing forms that are available in ACT were developed in *Tracker*. The use of *Tracker* was found to substantially reduce the knowledge engineering and GUI design stages of the ACT development. The knowledge engineer was able to use *Tracker* to rapidly author a set of forms that mimic the paper forms used by the target end users at the Air Force Air Mobility Command (AMC). *Tracker* was used to interactively develop several forms/templates with select subject matter experts and end users. This provided quick feedback on form design. Using *Tracker* we were able to demonstrate to the end users how software agents in ACT could be used to propagate data values among related forms and to enforce constraints on information element specification; e.g., date-time specification. By using *Tracker* we were also able to quickly develop some prototype forms-based GUIs, presenting them to the end users and subject matter experts for review during knowledge acquisition sessions. Once the data entry form designs were accepted by the end users, we were able to use *Tracker* to export the developed form(s) to ACT for continued integration with ACT ontology and software-agent modules. Because of the success of this approach, we added a tool bar to ACT to directly support the import of *Tracker* developed templates [18].

7. General *Tracker* Discussion

7.1 Using XML

XML is an excellent method for organizing a lot of information. It incorporates clean and simple semantic labeling and a hierarchical structure that exactly captures how many pieces of data relate to each other. The structure that XML describes is a hierarchical tree, exemplified by the structure of a book: A book has one or more chapters, a chapter has one or more paragraphs, a paragraph has one or more sentences, a sentence has one or more words, a word has one or more letters. In a physical book, there is no sharing of any of the structure, although it is a certainty that most of the words will appear in all the chapters, and some sentences might get repeated.

A tree structure also describes other hierarchical relationships, such as the military chain of command and personnel rankings (and similar relationships in many other organizations); in fact, anything that involves taxonomy-like categorization fits a tree structure quite well.

Some information does not fit into a tree structure well, and for this kind of data, XML provides at best an approximate solution. To continue the book example, a sentence also has a subject, predicate and an optional object. But subjects, predicates, and objects (S/P/O) do not always occur in that order, so in some cases it might be important to capture the S/P/O structure separately from the text of the sentence, which leads to a need for sharing of some sort, via a workaround that is an application-specific description of how non-tree data is represented in a tree.

Information is often shared across related documents, or is logically connected to other documents: the World Wide Web is the largest and most well known non-tree-structured collection of data. Links can go from anywhere to anywhere else.

Tracker incorporates several linking strategies. Note that the behavior of a *Tracker* link is more complex than links on a web page. *Tracker* links actually show the linked values, in addition to the link. Individual data values can be aggregated without the user having to jump from one source to another.

Unlike other XML-based concepts, *Tracker* does not have, use, or require a Document Type Definition (DTD) or schema. This allows *Tracker* templates to be completely free form, and can result in semantically rich XML. It also allows *Tracker* to open any XML file. Note: *Tracker* will do a better job at presentation if the nodes have certain attributes that describe the presentation format, but none of them are required; a completely unspecified node will present as a simple JTextField if nothing else.

7.1.1 Xpath Work and Scripts

Xpath is a specification of, and Xalan is the code library for, retrieving individual node values from XML documents. It explicitly supports the tree-structured content that comprises an XML data file. Consequently it is ideal for *Tracker*'s use, and appears extensively in template field definitions.

Any template data field can have a value derived directly from another field in the same template or another template. XPath is the specification for searching the XML to find that specific value. There is a "wizard" tool in the "Modify Properties" dialog for each template field, which steps the template author through finding a field's Xpath. *Tracker* allows the same random (and possibly circular) connectedness that the Web allows. The author is not prevented from making a circular link chain, but there is no infinite loop possible in a computed field value unless the author introduces it; *Tracker* processes script fields one at a time, and they are all single-depth computations (i.e., an Xpath grabs a value, not a computation to perform).

7.1.1.1 Slaving Template Fields to Others

Tracker templates can refer via XPath to other fields in the same template or fields in other templates. When the field is in the same template, this is called *slaving*: one field's value tracks another field's value at all times. The *Tracker* interface provides multiple methods for setting and displaying slaved values.

7.1.1.2 Use of JScheme for Building Scripts

Early during this project it was clear that there were going to be data fields whose value would be a computed value, and that there would be infinite variation on what those computations should be; therefore *Tracker* needed a scripting language wired to the fields. At the same time, it needed to be something fairly compact, and have clean access into Java.

JScheme is the scripting language we adopted for *Tracker*. It is a Scheme RSR4 interpreter (minus continuations) written in Java. It also has full access to Java itself, and is capable of doing anything Java can do. This means the scripting language is quite powerful. A script for a field can be the simplest of code fragments: (+ 3 4), and the value "7" would be shown as the field value.

A script can be far more complex than that, referring via XPath to other fields in the template, testing values and branching on the results, combining values into a more descriptive result, etc. For example, some of the DC2S templates have scripts up to about 20 lines long, which refer to other fields, and produce a variety of answers on screen depending on what values are found. The possibility of a "formula" language, which would be a simpler form of scripting, was discussed off and on, but never implemented. For comparison, this would be like the formula language in Excel versus the Visual Basic scripting capability.

Rules and constraints are specialized variations of scripting. If-then rules could actually be implemented as very small templates; constraints probably could also, to an extent, but would probably lose some of their expressive power.

7.1.1.3 Use of JScheme to Support Collaboration

We also experimented with the use of JScheme to enhance our collaboration technique. The approach was to use the Jabber [19] pub/sub environment and JScheme to send messages in Jabber from inside *Tracker*. The idea was that each *Tracker* system would register to a Jabber server and be used as follows:

- *Template server* – One *Tracker* system is designated the template server. To view a template, it sends a message to subscriber *Tracker* systems containing the template. Each subscriber can send the template back once it is updated.

- *Collaboration* – Each *Tracker* user would join the active templates chat room, which would broadcast messages to everyone – similar to what *Tracker* does now.

- *Diverse tool access* – Various Active Template tools such as *Tracker* and D3i could exchange information and provide services to each other.

7.2 Use of Heterogeneous Data Sources

7.2.1 Databases

Databases are well-structured repositories of information. They are not hierarchical in nature (although an application-specific interface can make one appear to be so). Our experiments with *Tracker* indicated that access to the information stored in databases, either at the template level or at the slot level, was of critical importance to most users.

Tracker supports several database features that allow the user to read/write to a database. However, *Tracker* can also auto-generate a template that contains the data fields associated with a database. The auto-generated template can then be saved and used as the basis for further template developed, or generated as needed to get real time database updates.

There are several approaches to using databases in *Tracker* and those that have been implemented were driven by our experiments. The only requirement in *Tracker* for the built-in database assessors is that the database be accessible via JDBC/ODBC. In *Tracker,* a single data value can be read from a single record in a database table. An entire record can be read, and will appear to be a flat template (it is in fact converted to XML at read time, and the XML is then parsed into the equivalent template structure).

7.2.2 Map Files

Tracker has been developed to use the OpenMap mapping tool, and for our work with ARSOBL, to use the mapping tool called C2PC. Geographical locations, as well as graphical icons and sketches can be displayed directly from a template.

7.2.3 Datafiles

Most of the data in *Tracker* is stored in files. Users can create templates to support a particular problem-solving domain and create instances from those templates. Both the templates and the instances are generally stored as files, and can be managed much the same way as other files are managed. For the DC2S problem domain, we created special functions that supported the creation of a crisis action mission folder from a set of templates. The user could then manage the instances using the common PC operating system options.

7.3 Use of Websites

There are two possible uses of websites in *Tracker*. The first is as a repository of templates, just as the web is a repository of web pages. (This can include being a repository of things used in templates that are themselves not templates, such as images.) The second is as a source of other data, like a database, or an XML file of some sort that is going to have a value extracted.

Templates are intended to allow data to flow in both directions: open an empty one, enter data, and save the result. On the other hand, web pages are generally unidirectional, allowing data to flow from the server to the client (usually by means of a browser). This means that without some custom programming on the server side, *Tracker* cannot write a template back to a web site.

While a web page might contain interesting content (at least as it is visually presented in a browser), because not all web pages are written using XML-compatible code (XHTML), it may be impossible for *Tracker* to find such a page because *Tracker* requires the data to be formatted as XML. In addition, even XHTML might be so semantically empty that correctly extracting the correct value could be quite difficult, exceeding the ability of the XPath wizard. Therefore, any use of *Tracker* with a website-based data repository must take into account this requirement for semantically rich, XHTML-based web pages.

7.4 Use of Images and Graphics

If a data field is to contain an image as opposed to a text value, the XML node’s actual value is the pathname to that image file, but the presentation in the window is the image itself. There are six kinds of images possible: GIF, JPG, TIFF, BMP, PNG, and SVG (Scaleable Vector Graphics). The SVG image type is actually a vector picture, while the other types are bitmaps. GIF, JPG, and PNG can have a pathname value that is actually a URL, and load an image from the web. Image files for the DC2S templates were typically images of target areas on the ground, like satellite photos, or images of targets.

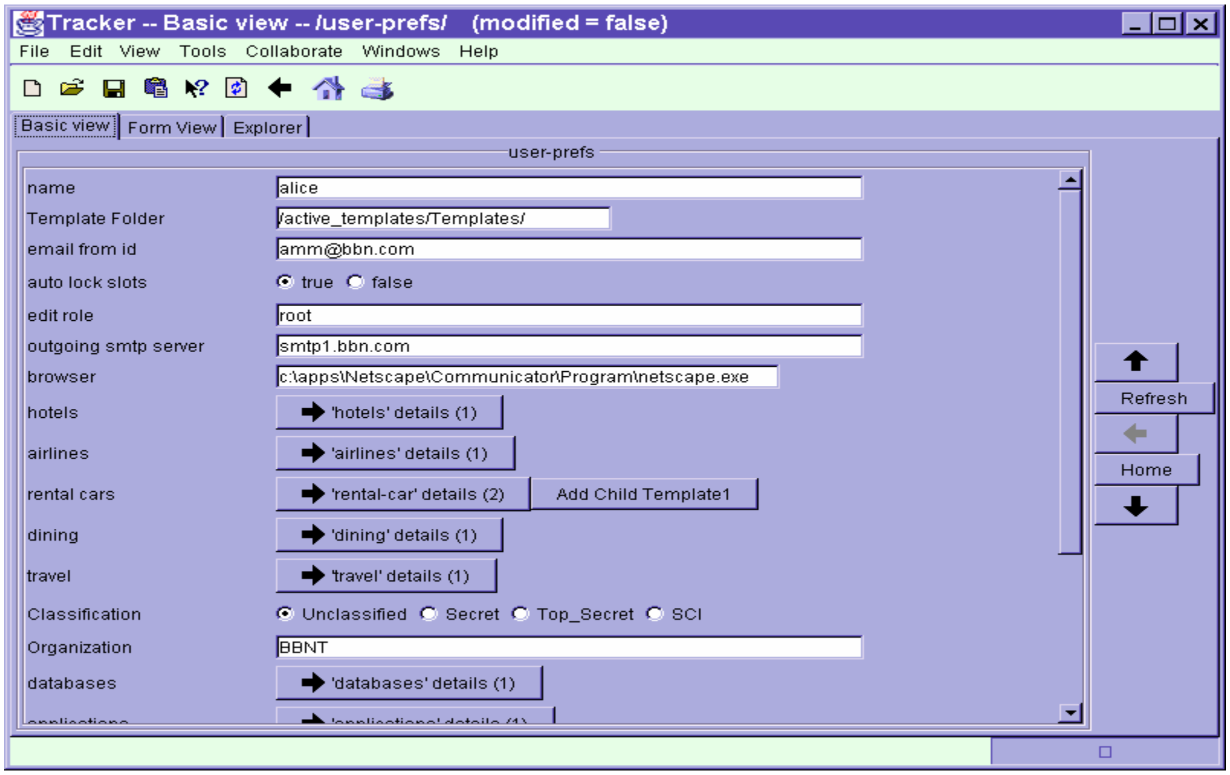
7.5 Domains

A *Tracker* "Domain" is an XML file (i.e., just another template) that captures aspects about a conceptual area common to all templates in that area, but is not explicitly part of any of those templates. It typically specifies script code files, Java code files, template-specific modifications to the menu-bar, domain-specific help, and the loading of custom widgets.

7.6 User Preferences

Web browser programs have a number of user-specific behaviors that the user can control that are independent of any and all templates. These are captured as User Preferences.

Tracker has the same concept, handled in essentially the same way, except that *Tracker*'s User Preferences are just another template (with a specific behavior unique to it). As just another template, the user-preferences (See Figure 24) can incorporate any amount of information that is not actually part of the built-in user preferences, and other templates could use that additional information. For example, the *Tracker* user preferences can specify preferred browsers, but also a preferred hotel.



name	alice
Template Folder	/active_templates/Templates/
email from id	amm@bbn.com
auto lock slots	[✓ true] false
edit role	root
outgoing smtp server	smtp1.bbn.com
browser	c:\apps\Netscape\Communicator\Program\netscape.exe
hotels	'hotels' details (1)
airlines	'airlines' details (1)
rental cars	'rental-car' details (2) Add Child Template1
dining	'dining' details (1)
travel	'travel' details (1)
Classification	[✓ Unclassified] Secret Top_Secret SCI
Organization	BBNT
databases	'databases' details (1)

Figure 27 - Example User Preferences in Tracker

7.7 Built-in Help

Tracker has built-in help like most modern software. Instead of a set of HTML help pages or an Acrobat file, *Tracker*'s help was written using *Tracker*, and is simply another set of templates. A number of screen-capture pictures are included in the Help system as examples.

7.8 Collaboration

Tracker provides several approaches to collaboration with templates. Each is discussed in this section.

7.8.1 Email

A template is a text file containing XML. As such, it is something that can easily be emailed between users. One user could fill in one part, email it to another user who could fill in another part, send it to someone else, etc.

This is a slow, asynchronous method, but nearly everyone has and can use email, so *Tracker* includes the ability to send a template as an email attachment to someone else.

Tracker includes a capability to create an address book for this purpose, but does not have a way to use or import an address book from another application.

7.8.2 Shared Drive/Server

In a domain with linked templates, all templates need to stay together in order for those links to work their best. Email is not an effective solution for this, because each template in effect lives alone.

One solution is to keep each group of linked instances together in the same folder all the time, but this means that all users of those templates must have access to that folder. This leads to the creation and use of network-shared folders. Depending on the domain and how many instances are necessary, it is probable that some user groups will need to establish a shared folder explicitly for their use, and do some other customization of *Tracker* to support this effectively. This is what was done for the DC2S domain implementation. There are a number of sub-folders that break down Mission Planning into Crisis and Deliberate, and break those two down into Direct Action (DA), Foreign Internal Defense (FID), Surveillance and Reconnaissance (SR), and Unconventional Warfare (UW). In collaboration experiments with DC2S, the entire folder structure was shared over the network.

7.8.3 Peer-to-Peer Live Sharing

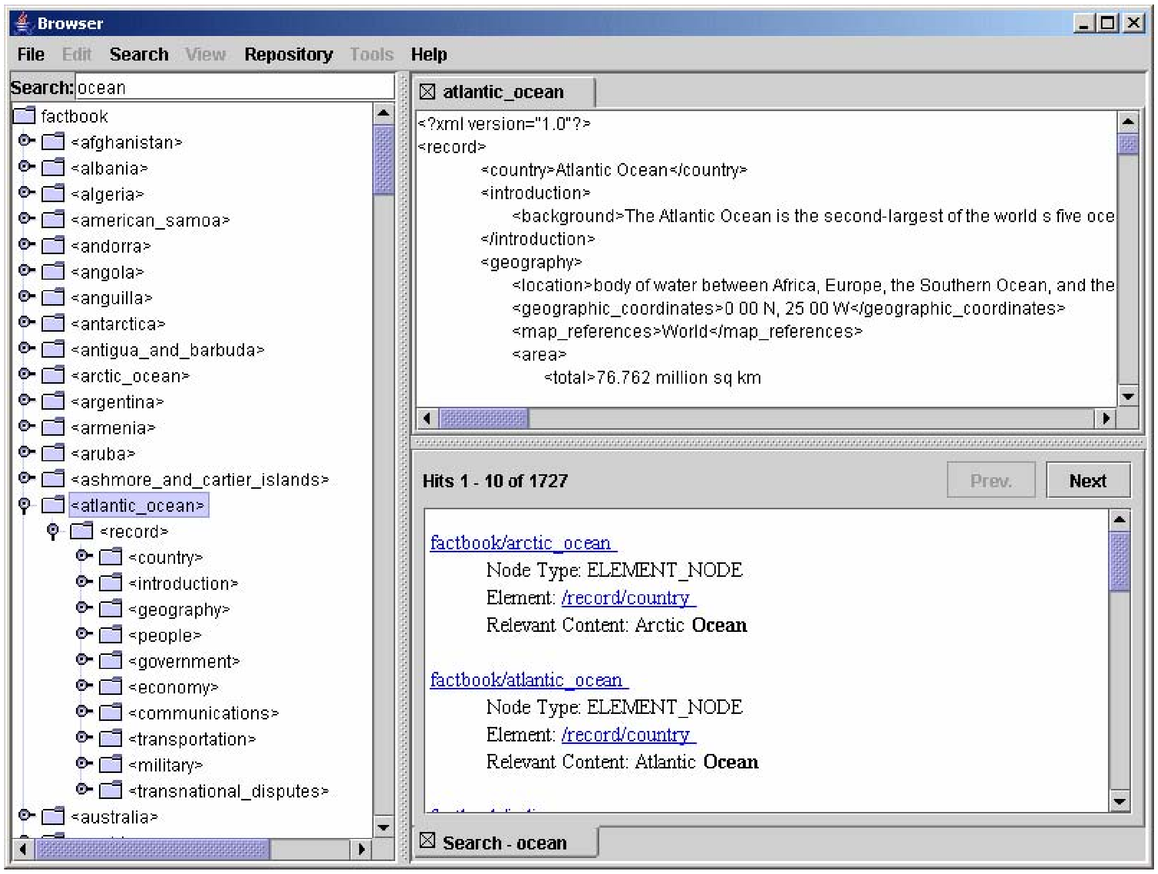
Tracker's collaboration session has more in common with network-based games than it does with something like NetMeeting. In NetMeeting, only one person can have control over the session at a time because pixels are being shared rather than data content. In a network game, only data is shared because in general those kinds of games are high-speed action games where all players have to act/react simultaneously, and the overhead for network traffic any greater than the few bytes necessary to state position and motion vector is too much. A live collaboration session in *Tracker* is based on a hub/spoke client-server implementation. One machine acts as "meeting host", the others connect to it as participants. Any one can open a template and share it. That template is then available to everyone else to be opened and worked on.

Values for data fields are transmitted from the user making a change to all other users. This way, all users are kept current with the latest entries. Everyone can work on the same template at the same time—fields are locked at the individual field level, and the lock is released when that user “changes the input focus”, which means clicking on some other GUI widget.

8. The Browser

There are two versions of the “browser” that were developed to support *Tracker*. One is the browser that is displayed in Figure 9. That browser can be called from inside *Tracker* and only works with *Tracker*. It is lightweight and very useful for managing templates and instances that have been developed using *Tracker*.

However, in our experiments with other Active Templates tools we found that we needed a more powerful browser, one with more advanced search features that would allow the user to construct a repository using XML files from any source. For example, Figure 28 shows the interface of the more powerful browser. Note that a query has been entered in the Search option. This particular repository is composed of the CIA Fact Book country data. The browser will highlight the first match in the pane on the left, and specify the number of hits in the pane on the lower right. The user can choose to view the returns in detail. In this case, the user is looking at detailed data about the “Atlantic ocean”. The tool also supports advanced search, such as a search for a phrase or for a complicated match expression.



```
<?xml version="1.0"?>
<record>
        <country>Atlantic Ocean</country>
        <introduction>
            <background>The Atlantic Ocean is the second-largest of the world s five oce
        </introduction>
        <geography>
            <location>body of water between Africa, Europe, the Southern Ocean, and the
            <geographic_coordinates>0 00 N, 25 00 W</geographic_coordinates>
            <map_references>World</map_references>
            <area>
                <total>76.762 million sq km
```

Hits 1 - 10 of 1727

factbook/arctic_ocean
    Node Type: ELEMENT_NODE
    Element: /record/country
    Relevant Content: Arctic **Ocean**

factbook/atlantic_ocean
    Node Type: ELEMENT_NODE
    Element: /record/country
    Relevant Content: Atlantic **Ocean**

Figure 28 - Browser with Query for Ocean against the XML Version of the CIA Fact Book

The stand-alone Browser was developed to search a collection of documents for items relevant to some set of query terms. Search hits might reference entire XML documents or individual XML elements relevant to the query. In Figure 28, the search for “ocean” returns several XML country documents as well as individual XML elements such as the text in the “background” for the Atlantic Ocean. In developing the browser we incorporated information retrieval technology currently used by web search engines such as Google, Yahoo, and Teoma.

The browser search capability was implemented using the Lucene open-source search engine developed under the Apache Jakarta Project. Lucene supports full-text search, and its plug-in based architecture makes it adaptable to a variety of applications. The Lucene search engine employs a vector space retrieval model with a similarity function based on term frequency and inverse document frequency metrics. For a description of these metrics, see Jockman, et al. [20].

The application of Lucene to searching XML documents in the browser’s repository was adapted from work published by Jockman, Kimber, and Reynolds of Isogen International. This section outlines the integration of Lucene with the stand-alone browser application and includes discussions of the document-indexing scheme, the fields indexed for each document, an optimization applied to the indexing process, and some search features.

8.1 Indexing Scheme

A Lucene Document contains a collection of fields, each consisting of a name and a value. The value portion of each field contains the data to be indexed. A Lucene index contains all unique Terms extracted from the value portion of each field in each Document that has been indexed. Each Term in the index contains references to the Documents it appeared in, the number of times it appeared in each document, and the position at which it appeared on each occurrence. The textual data of each field is passed through an Analyzer that enforces a policy for extracting Terms. The Analyzer implemented for the browser search application performs several filter operations. The Analyzer starts by removing apostrophe characters from possessive constructs and periods from acronyms in order to normalize the text for the next step. Then the Analyzer converts the characters in each word to their lowercase form and filters certain words based on a "Stop Words" list. The "Stop Words" list contains words such as "and", "the", and "but" that appear very frequently in text and are unlikely to be used as search criteria. The Analyzer applies the "Porter stemmer algorithm" [21][*] to extract the root of each word to be stored as a Term in the index.

The key to adapting Lucene to operate on a specific document type is defining how that particular document type will be represented as (or decomposed into) Lucene Document objects. In order to take advantage of the structure inherent to XML documents it was decided that each document would be indexed at the element level. A DOM parser is used to extract each individual element from an XML document and index it as a separate Lucene Document. In order to capture relationships among elements within the same document, the full content of each document is also indexed as a Lucene Document. The fields extracted from each element are described in the next section.

8.2 Fields

This section describes the fields that are indexed for each document and associated XML elements added to the browser's repository.

8.2.1 Node Type

The node type field indicates the type of content associated with each Lucene Document. If the Lucene Document represents the content of an entire XML document added to the repository, then the node type is set to "ALL_CONTENT". If the Lucene Document represents the content of a single XML element, then the node type field is set to "ELEMENT_NODE". This field is displayed in the search results to indicate whether a hit occurred on a single element or on a full document. If the indexing scheme were extended to separate XML content into additional DOM node types, then this field could be used to indicate the node type represented by each Lucene Document.

[*] Stemming is the process of applying an algorithm to extract the root word of a search term. The root word can then be used to derive additional search terms. For example, it may be desirable for the search engine to return documents containing the term "stemming" or "stemmed" if the user enters the term "stemmer". The browser search implementation makes use of the "Porter stemmer algorithm" when indexing documents such that all variations on the same root word get linked to the same term in the search index. The same stemming algorithm is also applied to all search terms of a query.

8.2.2 Collection

The collection that a document belongs to is included as a field in each Lucene Document derived from the XML document. This supports structuring a search on documents in one or more collections. The basic search function available from the browser frame appends "AND collection:<current collection>" to the entered query to restrict results to those that belong to the collection that is currently loaded in the browser. The *advanced search interface* currently supports selecting one collection or all collections to establish the context for a particular query. A user familiar with the query syntax could construct a query that considered documents in any combination of collections.

8.2.3 Tagname

If the Lucene Document represents an XML element, the tagname field contains the name of the tag enveloping the element. The tagname field supports limiting a search to only the elements of a certain type. The *advanced search interface* includes a "with the TAG" field for limiting a search in this manner. Use of the "with the TAG" field translates to adding " AND tagname:<user_supplied_text>" to the other search terms in the query.

8.2.4 Attributes

If the Lucene Document represents an XML element, the attributes field contains the name and value pairs of each attribute included in the element. A space character is inserted between the name and value of each attribute so that the tokenizer that feeds the indexer with words does not treat the attribute as a single term (i.e., the "=" character that typically delimits a name and value is replaced with a space). The attributes field allows a user to limit a search to elements containing a specific attribute name and/or value. The "Advanced Search" interface includes a "whose ATTIBUTES contain" field for limiting a search in this manner. Use of the "whose ATTRIBUTES contain" field translates to adding "AND attributes:<user_supplied_text>" to the other search terms in the query.

8.2.5 Content

Depending on the "Node Type", the content field contains either all text included in an XML document or all text between the start and end tags of an XML element. The content field is the field that is searched by default in all queries executed from the various search interfaces in the browser. In the case of an XML element, the content field also contains the contents of the tagname and attributes fields as well as the tagnames of each ancestor element. The content field in each Lucene Document also contains the name of the repository document from which it was derived. If the document name contains information that distinguishes it from other documents (which is usually the case), then it makes sense to support limiting a search to the elements within a single document by including the document name as one of the search terms.

8.3 Optimization

The Lucene search engine relies on a file-based index. While the developers of Lucene have produced a file-based scheme that is efficient for search tasks, producing the index can be very I/O intensive. That is, every document indexed by the system results in several "writes" to a file. Our indexing scheme significantly taxed the system because we index every element of an

XML document as if it were a separate document. To mitigate this, we added a memory buffer to the indexer such that several documents could be indexed (and stored in memory) before performing a write to the index file. The indexer is currently configured to support 10000 "documents" (i.e., documents and document elements) in the memory-based index before updating the file-based index. This optimization resulted in a significant decrease in the time required to index a collection of XML documents. For example, the CIA Fact Book collection took about 30 minutes to index prior to the optimization. After the memory buffer was added the indexing time decreased to just less than 3 minutes.

8.4 Search Features

One of the goals for the search component was to incorporate some of the technologies currently employed by some of the more popular web search engines (e.g., Google, Yahoo, Teoma, etc.) to improve the search experience. This section provides some detail on a few of the features included in the browser search mechanism.

8.4.1 Find Similar

The "Find Similar" feature is available as a hyperlink option on each of the hit descriptions displayed in the search results. "Find Similar" attempts to extract the terms that most uniquely identify a document and then construct a query based on these terms. The query is constructed by connecting the top terms with the OR operator. The top terms are derived by sorting the list of document terms in descending order by the weight of each term. The weight is computed by multiplying the normalized term frequency by the inverse document frequency. This sorting scheme attempts to identify the terms that appear most frequently in the selected document while appearing least frequently when considered across all documents. The "Find Similar" implementation currently selects the top five keywords from which to form a new keyword.

The main drawback to the current implementation of "Find Similar" is that it often results in a large number of hits. One of the contributors to this problem is the use of the OR operator to connect the derived search terms. Only one of the search terms has to be found in a document for it to be considered "similar" – this is not quite ideal; however, using the AND operator is not likely to return many (if any) results. Another contributor may be the inclusion of the filename in the content file of each indexed document. Because a filename only exists in documents derived from the file having that name, it is likely that the filename will bubble up as one of the terms that make the document unique among all documents in the repository. When a query is constructed using the filename as one of the search terms, the search engine generates a hit for every document/element derived from that file.

The "Find Similar" feature seems to work best on very distinct elements, for example, running the routine on the HIV infection rate element of a country in the CIA Fact Book retrieves the same element for all other countries. The current implementation should be considered experimental.

8.4.2 Did You Mean

The browser search capability includes a "Did You Mean" feature that attempts to catch misspellings by suggesting alternative search terms derived through a multi-stage heuristic. For each search term entered by the user, the search engine computes the Levenshtein distance [22] against all terms currently in the index. Potential alternatives must appear in at least one document and must have at least the first two letters in common with the word that the user entered. Of the terms that meet these requirements, the one with the smallest Levenshtein distance is reported as a possible alternative to the user.

The process used to find alternate terms is equivalent to the query expansion that occurs in a fuzzy search. If nothing is found in the index, then a dictionary of common terms is consulted to find possible alternatives. The browser currently utilizes the Jazzy open-source spell-checking library [23] to perform the dictionary check. Jazzy also makes use of the Levenshtein distance to suggest possible alternate terms.

9. Conclusion

Like *Tracker*, many of the AcT tools [24] were developed while XML and tools to create, edit and work with XML were also being developed by commercial vendors. Although some of the capability that *Tracker* was developed to provide are now readily available in tools like XML-Spy (http://www.altova.com/), the intent of the *Tracker* tool was to use forms to dynamically support real-time problem solving.

Forms-based tools abound in many environments and are primarily used to collect data. What we have demonstrated with *Tracker*, and with many of the other AcT tools, is that by making the forms “active” additional computational power is provided. For example, today in many command centers planners enter data in one form and then manually re-enter it into the PowerPoint briefs and other military command documents as needed. With the DC2S experiment we saw that the PowerPoint briefs can be created *automatically* using the planning data. As the planning environment changes, so do the briefings. In the IWARN experiment, we saw that by using *Tracker* to create the EAR forms, the end user could have the flexibility to add new fields to describe some unanticipated anomaly associated with a pre-defined incident. In this way, the forms would be able to grow and mature to meet the ever-changing needs of a dynamic environment.

Creating forms can be tedious, but with the authoring capability of *Tracker* we have been able to quickly develop forms-based interfaces for other applications such as the ACT tool described in this paper. Although we never really adapted *Tracker* to handle OWL data, we did demonstrate that by referencing an OWL ontology while developing the fields of a *Tracker* template, any application using those OWL ontologies also could quickly use the templates. In retrospect, one of the most powerful aspects of *Tracker* was the fact that it could be used to create forms-based interfaces that required an XML structure.

Although *Tracker* and other AcT tools like CommandLink could be used to develop and manage workflows, the only actual usage of this capability was through the CAP Checklist that was developed to support the SOF DC2S application.

Finally, our original intent was to develop a tool where we could dynamically link-in templates that were stored somewhere, and rapidly use them in a new context. The experiments with D3i and the resulting development of the Browser were examples of this capability. Currently the Browser is being used to support repository development in the DARPA JAGUAR program, and it is in that context that we continue to pursue the linking of templates and instances with each other.

10. References

[1] Java: http://www.sun.com/java/

[2] TurboTax: http://www.turbotax.com/

[3] Mulvehill, A., Reilly, J., and Krisler, B. “Flexible Data Entry for Information Warning and Response Systems”, presented at the International Command and Control Research and Technology Symposium (ICCRTS—05), June 2005.

[4] XpertRule: http://www.attar.com/index.htm

[5] Xpath/Xalan: http://www.w3.org/TR/xpath

[6] C2PC: http://www.globalsecurity.org/intell/library/reports/2001/compendium/c2pc.htm

[7] NetMeeting: http://www.microsoft.com/windows/netmeeting/

[8] JScheme: http://jscheme.sourceforge.net/jscheme/mainwebpage.html

[9] Ambite, J., Knoblock, C., Muslea, M., and Minton, S., “Conditional Constraint Networks for Interleaved Planning and Information Gathering”, IEEE Intelligent Systems, pp. 25-33, March/April 2005.

[10] Dyer, D., D3i Software, http://activecomputing.org/prototypes.html

[11] XForms: http://www.w3.org/TR/2003/REC-xforms-20031014/

[12] InfoPath: http://office.microsoft.com/en-us/FX010857921033.aspx

[13] DAML: http://www.daml.org/

[14] Smith, M. K., Welty, C., and McGuiness, D. L., eds., “OWL Web Ontology Language Guide,” http://www.w3.org/TR/2004/REC-owl-guide-20040210/. 2004.

[15] Nau, D., Au, T., Ilghami, O., Kuter, U., Munoz-Avila, H., Murdock, J., Wu, D., and Yaman, F., “Applications of SHOP and SHOP2”, IEEE Intelligent Systems, pp. 34-41, March/April 2005.

[16] Breslow, L., and Aha, D., NaCoDAE: Navy Conversational Decision Aids Environment. Technical Report AIC-97-018, Navy Center for Applied Research in Artificial Intelligence, Naval Research Laboratory, Washington, DC, 1997.

[17] Mulvehill, A., Benyo, B., Rager, D., and DePalma, E., “ACT – The Automated Clearance Tool: Improving the Diplomatic Clearance Process for AMC”, 2004 Command and Control Research Technology Symposium, June 2004 (Conference Proceedings)

[18] Mulvehill, A., “Authoring Templates With *Tracker*,” IEEE Intelligent Systems, pp. 42-45, March/April 2005.

[19] Jabber: http://www.jabber.org/about/overview.shtml. Also see, Moore, D., Wright, W. *Jabber Developer's Handbook*, 2004.

[20] Jockman, Brandon, Eliot Kimber, and Josh Reynolds. *Enabling Low-Cost XML-Aware Searching Capable of Complex Querying*. Isogen International, 2002.

[21] Baeza-Yates, R, and Ribeiro-Neto, B., *Modern Information Retrieval*, New York: Addison-Wesley, 1999.

[22] http://www.levenshtein.net/

[23] http://jazzy.sourceforge.net/

[24] Dyer, D., Cross, S., Knoblock, C., Minton, S., and Tate, A., "Guest Editors' Introduction: Planning with Templates", IEEE Intelligent Systems, pp. 13-15, March/April 2005.

11. Acknowledgements

We would like to thank Doug Dyer, the DARPA sponsor, AFRL and the Subject Matter Experts who supported the development of *Tracker* and of the experiment applications. This work was supported by DARPA and AFRL under Contract No. F30602-00-C-0039.

12. Appendix A

12.1 Example Communication Protocol between *Tracker* and HICAP

Codes for the communications between *Tracker* and HICAP

1. *Tracker*:

Sending:

Ta	Tells HICAP whenever any template is opened
Tb	Tells HICAP whenever any template is closed
Tc	Tells HICAP whenever any slot is updated, providing the new value
Td	Tells HICAP when user accepts or rejects a suggestion

Receiving (Maintains a pending/accepted/rejected window):

Te	Receives HICAP's suggestion for opening a specific template
Tf	Receives HICAP's suggestion for substituting a specific template
Tg	Receives HICAP's suggestion for a specific slot value
Th	Receives HICAP's suggestion; displays a given text message

2. HICAP:

Receiving:

Ha	Receives info on template openings
Hb	Receives info on template closings
Hc	Receives any change to any subscribed slot - HICAP checks whether this triggers any case
Hd	Receives acceptance/rejectance feedback on its suggestions

Sending:

He	Offer suggestions to *Tracker* for opening a specific template - HICAP displays its reasons in favor of this new template
Hf	Offer suggestions to *Tracker* for substituting specific template - HICAP displays its reasons in favor of this new template and against the currently-opened template
Hg	Offer suggestions to *Tracker* for a specific slot value - This might later be broken into two codes (ala Hf and Hg)
Hh	Offer other suggestions to *Tracker* (i.e., arbitrary text message) - Semantics of these suggestions will not be interpreted by *Tracker*

12.2 *Tracker*/HICAP Demo 1

1. User begins tax domain session
   - User opens whether-to-file template (Ta,Ha)
     - Causes HICAP to open WhetherToFile.cdf (Goal: DetermineWhetherToFile)
2. User inputs whether-to-file slot values (Tc,Hc):
   - Filing Status (FilingStatus): Married-filing-jointly
   - Gross Income (grossIncome): $3,604 (i.e., x<$12,950)
   - Age at end of 2000 (ageEnd2000X): 39 (i.e., x<65)
   - User pushes "Update Templates" button
3. HICAP suggests filing no tax form (Hh,Th)
   - Suggestion = #1; *Tracker* pushes this onto PendingList
   - Case: RuleFileNoForm in WhetherToFile.cdf
4. User edits (one digit for) a whether-to-file slot value (Tc,Hc):
   - Gross Income (grossIncome): $43,604 (i.e., $12,950<x<$50,000)
5. HICAP suggests filing some tax form (Hh,Th)
   - Suggestion = #2; *Tracker* pushes this onto PendingList
   - Case: RuleFileSomeForm in WhetherToFile.cdf
6. User Edits Suggestion Lists:
   - User rejects Suggestion #1 (Td,Hd)
   - User accepts Suggestion #2 (Td,Hd)
   - User opens HiCAP*Tracker*2-select-tax-form template (Ta,Ha)
     - Causes HICAP to open SelectTaxForm.cdf (Goal: SelectTaxForm)
7. User inputs HiCAP*Tracker*2-select-tax-form slot values (Tc,Hc):
   - Received Dependent Benefits (receivedDepBenefits): No
   - Student Loan Interest Deduction (studentLoanInterest): No
   - Taxable Interest (taxable-interest): $168 (i.e., x<$400)
   - User pushes "Update Templates" button
8. HICAP suggests filing tax form 1040EZ (He,Te):
   - Suggestion = #3; *Tracker* pushes this onto PendingList
   - Case: RuleFileForm1040EZ_2 in SelectTaxForm.cdf
9. User Edits Suggestion Lists:
   - User accepts Suggestion #3 (Td,Hd)
   - User opens IRS-1040-EZ template (Ta,Ha)
     - Causes HICAP to open nothing

   Contents of IRS-1040-EZ template at this time:
   - Taxpayer: Clintonius Bush
     Taxpayer-SSN: 123-45-6789
     Street-Address: 1 CIA Way
     City: Arlington
     State: Virginia
     ZIP-code: 22217
     Spouses-name: Clintonia Bush

  Spouses-SSN: 123-45-6790
- Filing-Status: "Married-filing-jointly"
- taxpayer-pres-campaign: "No"
- spouse-pres-campaign: "Yes"
- Line 2 (taxable-interest): $168

10. User inserts the following into template IRS-1040-EZ (Tc,Hc):
    - Line 1 (Wages-salaries-tips): $42,836
11. User edits (one digit for) a IRS-1040-EZ slot value (Tc,Hc):
    - Line 2 (taxable-interest): $768 (i.e., x>$400)
    - User pushes "Update Templates" button
12. HICAP suggests substituting template IRS-1040 for IRS-1040-EZ (Hf,Tf)
    - Suggestion = #4; *Tracker* pushes this onto PendingList
    - Case: RuleFileForm1040_3 in SelectTaxForm.cdf
13. User edits Suggestion Lists:
    - User accepts Suggestion #4 (Hd,Td)
    - User closes Template IRS-1040-EZ (Tb,Hb)
    - User opens Template IRS-1040 (Ta,Ha)
      - Causes HICAP to open:
        - WhetherToFile2441.cdf
        - WhetherToChangeDeduction.cdf
    - Contents of IRS-1040 template at this time:
      - Taxpayer: Clintonius Bush
        Taxpayer-SSN: 123-45-6789
        Street-Address: 1 CIA Way
        City: Arlington
        State: Virginia
        ZIP-code: 22217
        Spouses-name: Clintonia Bush
        Spouses-SSN: 123-45-6790
      - Filing-Status: "Married-filing-jointly"
      - Exemptions:
        - Line 6a (Yourself): Yes
        - Line 6b (Spouse): Yes
        - Line 6c (Dependents):
          - Danius H. Bush
          - 111-11-1111
          - Son
          - Yes
        - Line 6d (total-exemptions): 3
      - Line 7 (Wages-salaries-tips): $42,836